Final Report

Title: Broadcast based control of multi-agent systems for consensus

AFOSR/AOARD Reference Number: AOARD-08-4098

AFOSR/AOARD Program Manager: John Seo, Lt Col, USAF
.

Period of Performance: 2008 - 2010

Submission Date: December 2010

PI: Debasish Ghose, Indian Institute of Science, Bangalore, India

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
08 DEC 2010	**FInal**	**01-06-2008 to 01-05-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Broadcast based control of multi-agent systems for consensus	**FA23860814098**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Debasish Ghose	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Indian Institute of Science,Dept. of Aerospace Engineering,Bangalore 560-012,India,IN,560-012	**N/A**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
AOARD, UNIT 45002, APO, AP, 96337-5002	**AOARD**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
	AOARD-084098

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research investigated the consensus problem where there is a centralized decision-maker which conveys action plans to the swarm members with the constraint that the action plan is broadcast to the members of the swarm, as against individual control actions by each agent to 1) Formulate the problem as one in which the central decision-maker has information about the states of the swarm members, 2) Determine a strategy for the decision-maker so that it can broadcast an action plan to all the swarm members at some frequency to bring about a consensus among the swarm members by having them implement the broadcast instruction, 3) Explore new solution for eliminating the known drawbacks(non-achievement of consensus and large computation time) using a linear programming based algorithm approach, 4) Refine this strategy to achieve quick convergence for large swarms, and 5) develop analytical convergence proofs and study the effect of varying number of swarm members and different randomness mechanisms on convergence.

15. SUBJECT TERMS

Guidance, Control System, Decision Making, systems engineering, cybernetics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**49**	

Objectives

There has been considerable interest in recent times in achieving consensus in swarms of agents using simple strategies. Work in this area has ranged from completely autonomous agents to partially autonomous agents, with several different frameworks for communication and information. In this proposal we intended to address the consensus problem where there is a centralized decision-maker which conveys action plans to the swarm members with the constraint that the action plan is *broadcast* to the members of the swarm, as against individual control actions by each agent. This implies that each member of the swarm receives exactly the same action advice and implements the same. However, the system is not autonomous in the sense that the information about the agents' states is available to the centralized decision-maker through its own sensing devices. We formulate the problem as one in which the central decision-maker has information about the states of the swarm members. This information is assumed to be obtained by the decision-maker using its own sensor suite. On the other hand, the swarm members have no global information. Their information is local and is restricted to a reduced state in their own local co-ordinate system. They are capable of receiving instructions through broadcast and implementing them in their local reference frame. The problem we address is to determine a strategy for the decision-maker so that it can broadcast an action plan to all the swarm members at some frequency. The objective is to bring about a consensus among the swarm members by having them implement the broadcast instruction. A further objective was to design a strategy that would require minimal computational time.

.

Status of effort

In our work we were able to design a strategy based on a randomization algorithm and a simplified linear programming formulation to achieve positional consensus among a fairly large number of agents using a reasonably small computational effort. Several strategies were examined and algorithms developed and evaluated and the results have been reported. In addition we have also designed a consensus algorithm in which each agent takes its decision autonomously. We have shown that the computational time advantage is obtained even for this case if we resort to a linear programming formulation instead of a the standard non-linear formulation that researchers have used earlier.

Abstract

This work addresses the problem of achieving rendezvous in a multi-agent system under various information paradigms. We consider two classes of algorithms (i) Broadcast based algorithms and (ii) Distributed control algorithms. In the first we consider both the centralized and decentralized cases. In the centralized case each agent is homogeneous and all agents are controlled by the same broadcast command from a centralized controller. This method has low communication cost. In the decentralized case each agent computes its control using the information it obtains from its neighboring agents and shares its control with its neighbours through a broadcast command. In the second case we consider each agent to implement its own control based on information gathered from its neighbours through a limited sensing capability. We show that in the distributed control algorithm, a modification in the decision domain of the agents yields significant benefits in terms of computational time, when compared with standard algorithms available in the literature. Moreover, we also show its straightforward application to higher dimensional problems which is a considerable improvement over available algorithms in the literature.

Some recent work in the literature, using ideal deterministic models for mobile robots, with its model based on an actual MEMS micro-robot, has shown that it is possible to achieve point convergence in this framework for two robots and to achieve limited consensus. But perfect positional consensus cannot be obtained for larger number of robots. In the literature, an optimization problem was formulated that minimizes the maximum distance between swarm members. However, the formulation and the solution suffer from two important drawbacks. One is that the swarm members cannot achieve point convergence (or perfect consensus) even after repeated application of the strategy. They can only be brought within a

certain distance of each other and no closer. The second drawback is that the optimization is rather time intensive and needs non-standard optimization techniques and software such as SOCP (second order cone programming) to obtain a solution. In our work we propose a solution for both the above drawbacks. In the case of the first drawback (that is, non-achievement of consensus) we propose a strategy that has a randomness function built into the broadcast command that will eventually help the swarm to achieve point convergence without relaxing the broadcast command paradigm of the original problem. In the second case (that is, large computation time), we will propose a LP (linear programming) based algorithm that is less computationally intensive than the SOCP based algorithm.

Personnel Supported

Kaushik Das (PhD Research student)

P.B. Sujit (Post-doctoral visitor)

Joesph Thomas (Master's Student)

Gaurav Rattan (Project assistant)

Eshwaran Vijay Kumar (Project assistant)

Publications (See Attachments for the actual papers)

K. Das and D. Ghose: Randomization mechanisms based positional consensus in homogeneous multi-agent systems, *Proceedings of the IISc Centenary International Conference and Exhibition on Aerospace Engineering (ICEAE2009)*, Bangalore, India, May 2009, pp. 1327-1336.

K. Das and D. Ghose: Positional consensus in multi-agent systems using a broadcast control mechanism, *Proceedings of the American Control Conference (ACC'2009)*, St. Louis, USA, June 2009, pp. 5731-5736.

K. Das and D. Ghose: Multi-agent rendezvous algorithms under various information paradigms, *Proceedings of the International Conference on Intelligent Unmanned Systems (ICIUS''2010)*, Bali, Indonesia, November 2010.

K. Das and D. Ghose: Multi-agent rendezvous algorithm with rectilinear decision domain [Book Chapter], *Trends in Intelligent Robotics* (Eds. P. Vadakkepat and J.-H. Kim), Communications in Computer and Information Science, Vol. 103, Springer Verlag, Berlin, Germany, pp. 162-169, 2010.

K. Das and D. Ghose: Positional consensus of multi-agent systems using linear programming based decentralized control with rectilinear decision domain [Book Chapter], *Trends in Intelligent Robotics* (Eds. P. Vadakkepat and J.-H. Kim), Communications in Computer and Information Science, Vol. 103, Springer Verlag, Berlin, Germany, pp. 178-185, 2010.

Interactions

(a) Participation/presentations at meetings, conferences, seminars, etc.

INDO-US MAV Workshop, Bangalore, November 2008.

American Control Conference, St. Louis, USA, June 2009.

IISc Centenary International Conference and Exhibition on Aerospace Engineering (ICEAE2009), Bangalore, India, May 2009.

INDO-US Workshop on System of Systems Engineering, IIT Kanpur, India, October 2009.

International Conference on Intelligent Unmanned Systems (ICIUS''2010), Bali, Indonesia, November 2010.

FIRA RoboWorld Congress, Bangalore, India, September 2010.

(b) Describe cases where knowledge resulting from your effort is used, or will be used, in a technology application. **Not all research projects will have such cases, but please list any that have occurred.**

None

Inventions:

(a) List discoveries, inventions, or patent disclosures. (If none, report None.). None

(b) Complete the attached **"DD Form 882, Report of Inventions and Subcontractors." Completed**

Honors/Awards

None

Archival Documentation

Softcopies of papers listed above under publications are attached.

Software and/or Hardware (if they are specified in the contract as part of final deliverables)

Not Applicable

2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Positional Consensus in Multi-Agent Systems using a Broadcast Control Mechanism

Kaushik Das and Debasish Ghose

Abstract— **In this paper a strategy for controlling a group of agents to achieve positional consensus is presented. The proposed technique is based on the constraint that every agents must be given the same control input through a broadcast communication mechanism. Although the control command is computed using state information in a global framework, the control input is implemented by the agents in a local coordinate frame. We propose a novel linear programming formulation that is computationally less intensive than earlier proposed methods. Moreover, we introduce a random perturbation input in the control command that helps us to achieve perfect consensus even for a large number of agents, which was not possible with the existing strategy in the literature. Moreover, we extend the method to achieve positional consensus at a pre-specified location. The effectiveness of the approach is illustrated through simulation results.**

I. Introduction

The principle of using multiple agents is motivated by the idea that instead of using a highly sophisticated and expensive robots, it may be advantageous in certain situations to use a group of small, simple, and relatively cheap robot. The group of agents can be used to accomplish various tasks in different environment such as tactical operations, exploratory space missions, remote monitoring with mobile sensor networks, avoidance of collision and over-crowding in automated air traffic control, cleanups of toxic spills, fire fighting and cooperative search with unmanned air vehicles.

One of the problems that is of paramount importance in multi-agent systems is that of achieving consensus, that is, achieving identical values for some specified subset of the states of the agents. For instance, the agents may try to converge to the same direction of movement [1] after some time or they might want to converge to a point. Both are problems in achieving consensus. If we have a centralized system with perfect information then achieving consensus is a trivial matter, since the central controller can instruct each agent suitably to reach a common consensus point. However, if the communication system has a constraint on the number of messages that it can communicate, then one may opt for a broadcast protocol where the central controller will communicate simple and identical instructions to all the agents through a broadcast mechanism. We further impose the additional constraint that each agent can interpret the control command only in its local coordinate frame or local state space. Only the central controller has access to the global states of the system. Some of these constraint are common to other problems of a similar nature (for instance, see [2], [3], [4], [5]).

This problem was motivated by a recent paper by Bretl [6] where a control strategy for a group of micro-robots is developed to perform a useful task even when every robot receives the same control signal. The paper considers point robots with simple kinematics. It was shown that when there are only two agents, there exists a broadcast control command (that is, both agents receive identical instructions from the central controller) using which both agents can meet at the same location at the same time, for almost all initial conditions. However, if the number of agents is more than two, then the best that the agents can achieve is to come close to each other within a certain distance (measured by the radius of the smallest disc that contains all the agents positions), which is a function of the initial conditions. Bretl [6] formulates this problem as an optimization problem that minimizes the radius of the disc, and proposes a solution using the second order cone programming (SOCP) technique [8]. However, using this strategy the agents cannot be made to converge to a point. Once the solution of the SOCP is implemented, no further improvement is possible. Bretl's paper was in turn motivated by an interesting paper by Donald et al. [7] on the development of untethered, steerable micro-robots, where every robot receives the same power and control signal through an underlying electrical grid.

Our paper makes several specific contributions. The first is to propose a strategy that uses the basic Bretl's model with an additional randomization feature that allows large number of agents to achieve positional consensus or point convergence on repeated application of the algorithm without compromising the broadcast constraint on the control command. The second contribution is that our method can be extended to the case where the agents can be made to converge to any pre-specified point. The third contribution is to propose an optimization problem for this task that is based on a linear programming formulation. This allows standard and easily available software to be used for obtaining the solution. Moreover, this formulation also retains the property that the number of decision variables, whose values are to be communicated to agents, remains unchanged even when the number of agents increases. Finally, we also propose some interesting properties related to the positional formation of agents when the LP based strategy is applied iteratively.

It is worth noting that the randomization feature in the algorithm has some similarity with the random perturbation used in Viscek's model [1]. Vicsek et al. [1] propose a simple

K. Das is a Research Scholar and D. Ghose is a Professor in the Guidance, Control, and Decision Systems Laboratory in the Department of Aerospace Engineering, at the Indian Institute of Science, Bangalore 560012, India kaushikdas+dghose@aero.iisc.ernet.in

This project is supported by an AOARD/AFOSR grant.

but compelling discrete time model of n autonomous agents (points or particles) all moving in the plane with constant speeds but with different headings. Each agent updates its heading using a local rule based on the average of the headings of its neighbors plus some random perturbation.

The paper is organized as follows: In Section II we consider two agents and show that it is possible to move two agents to a common location using identical control. In Section III we formulate a linear programming problem for minimizing the proximity between agents by using identical broadcast control. In Section IV we have discussed some results on the formation of the agents after the linear programming solution is implemented. In section V we introduce the notion of iterative solution of the problem by repeated use of the LP algorithm and show that introducing a random perturbation in the broadcast mechanism leads to point convergence of the agents by repeated application of the LP technique. In Section VI we present a modification of the algorithm to ensure that the swarm of agents converge to a pre-specified point. In section VII we show several simulation results that illustrate the salient features of the proposed algorithm. Section VIII concludes the paper with a discussion of possible future directions of research.

II. Formulation and Solution for Two Agents

We will first pose the problem in a general framework and then address the two agents case to clarify many of the assumptions and concepts discussed in the previous section.

Assume that n agents are located on an obstacle-free plane. We assume that the central controller has access to the global state of the system which, in this case, consists of the position ($x_i \in \mathbb{R}^2$) and orientation ($\theta_i \in (-\pi, \pi]$) of the agents, $i = 1, \ldots, n$. The central controller computes a common local control for the agents and broadcasts it to the agents for implementation. The local control is in the form of a tuple (θ, d), which is interpreted by each agent in its local frame of reference. Here, θ refers to the angle by which each agent changes its orientation, and d is a scalar that refers to the distance by which each agent moves after effecting the orientation change. Note that, the broadcast mechanism (θ, d) is the same for all the agents. Also, the local frame of reference for each agent is centered at the agent's location and its reference axis is oriented along its current orientation. As an illustration see Fig. 1, where agents are shown located initially at x_{i0} with initial orientation θ_{i0} in the global reference frame. If the control command broadcast to all the agents is (θ, d), then the agents implement it in their local coordinate frame by each of them changing their orientation by the same angle θ and advancing by the same distance d to reach the final destination x_{if}. Even in this figure it can be seen that by doing this the agents have come closer to each other. Our objective is to determine a (θ, d) such that the agents can achieve the closest proximity with each other.

Theorem 1: For two agents, for all initial conditions of the agents except when $\theta_{10} = \theta_{20}$, there exists a control (θ, d) using which point convergence can be achieved.

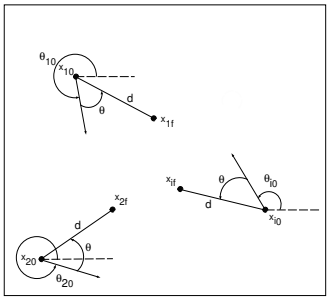


Fig. 1. Basic configuration

Note that this result is also available in Bretl [6] and is stated here for completion. The above theorem shows that it is possible to use a broadcast control command to make two agents meet at the same location simultaneously for almost all initial conditions. However, the solution is also unique and hence the location of the meeting point cannot be chosen arbitrarily. One can also interpret this result by noting that the final meeting point is on the Voronoi edge (equidistant line) between the two initial positions of the agents. It can be shown that only one unique point on the Voronoi edge satisfies the requirement that the orientation change angle is the same for both the agents (see Fig. 2). The point p moves on the equidistant line from $-\infty$ to $+\infty$ and the corresponding orientation angle change θ is plotted for the two agents. The intersection of the two curves is the unique control command point. It can be seen that when the number

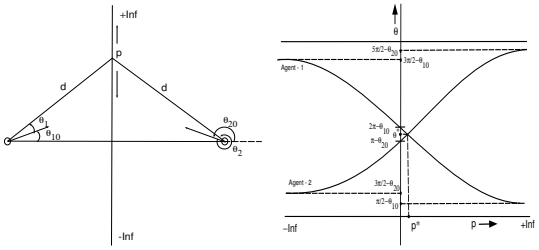


Fig. 2. Voronoi interpretation

of agents is more than two, each pair gives rise to a different unique meeting point. Thus, there does not exist a common control command to be broadcast so that all the agents meet at a point. In the absence of such a command, the best that can be done is to determine a (θ, d) which brings the agents in closest proximity with each other. Note that in this case (θ, d) may not be unique.

In the next two sections we will propose solutions to overcome both the drawbacks without compromising the broadcast based control mechanism.

III. A Linear Programming Formulation

Let the initial position and initial orientation of the n agents be $x_{i0} = (p_{i1} \;\; p_{i2}) \in \mathbb{R}^2$ and $\theta_i \in (-\pi, \pi]$, respectively, for all $i \in \{1, ..., n\}$. As before, we define the control command to be broadcast as (θ, d). We define our performance measure as the half length, denoted by $r > 0$,

of the side of a square oriented along the global coordinate frame, and containing all the final positions of the agents. Let this square be centered at $z = (z_1, z_2) \in \mathbb{R}^2$.

Assuming that all the agents execute the command (θ, d), their final positions, given by $x_{if} = [q_{i1} \quad q_{i2}] \in \mathbb{R}^2$ will be,

$$x_{if} \quad = \quad x_{i0} + R(\theta_{i0})R(\theta) \begin{bmatrix} d \\ 0 \end{bmatrix} \tag{1}$$

That is,

$$\begin{bmatrix} q_{i1} \\ q_{i2} \end{bmatrix} \quad = \quad \begin{bmatrix} p_{i1} \\ p_{i2} \end{bmatrix} + \begin{bmatrix} \cos\theta_{i0} & -\sin\theta_{i0} \\ \sin\theta_{i0} & \cos\theta_{i0} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2}$$

where, $u_1 = d\cos\theta$ and $u_2 = d\sin\theta$ are the control variables that replace (θ, d). Note that Eqn. (2) are linear equations. Now, we formulate the linear programming problem as,

Minimize r

Subject to

$$-r \leq p_{i1} + u_1\cos\theta_{i0} - u_2\sin\theta_{i0} - z_1 \leq r \tag{3}$$

$$-r \leq p_{i2} + u_1\sin\theta_{i0} + u_2\cos\theta_{i0} - z_2 \leq r \tag{4}$$

$$i = 1, \ldots, n.$$

$$r \geq 0 \tag{5}$$

The above is a linear programming problem with the decision vector as (r, z_1, z_2, u_1, u_2). Note that the decision vector remains same irrespective of the number of agents. Only the number of inequality constraint increases with the number of agents. Also, note that z_1,z_2,u_1 and u_2 are free variables and can take both positive or negative values.

IV. Some Results on the Formation of Agents

After executing the LP the distance between i and j agent along x-axis and y-axis will be

$$\begin{aligned} d_{x_{ij}} &= (p_{i1} - p_{j1}) + d(\cos(\theta_i + \theta) - \cos(\theta_j + \theta)) \\ &= (p_{i1} - p_{j1}) + dC \end{aligned} \tag{6}$$

$$\begin{aligned} d_{y_{ij}} &= (p_{i1} - p_{j1}) + d(\sin(\theta_i + \theta) - \sin(\theta_j + \theta)) \\ &= (p_{i1} - p_{j1}) + dS \end{aligned} \tag{7}$$

where, $C = (\cos(\theta_i + \theta) - \cos(\theta_j + \theta))$, $S = (\sin(\theta_i + \theta) - \sin(\theta_j + \theta))$ and $i, j \in \{1 \ldots n\}$. Thus the resulting distance along x-axis and y-axis are dictated by the value of C and S.

After executing the LP, the new formation of the agent obtained is of interest. Below, we investigate some properties of the formation. For this let us define the span of the formation along X and Y axis as follows: Let (x_i, y_i) be the position of the agents, where $i \in I = \{1, \ldots, n\}$. Then define $x_{max} = \max\{x_i\}_{i\in I}$, $x_{min} = \min\{x_i\}_{i\in I}$, $y_{max} = \max\{y_i\}_{i\in I}$ and $y_{min} = \min\{y_i\}_{i\in I}$. Then the span of the formation along the X and Y axis are given by,

$$S_x = x_{max} - x_{min}$$

$$S_y = y_{max} - y_{min}$$

The formation is said to be square if $S_x = S_y$ and rectangular otherwise. Essentially, the spans are the length of the sides of the minimal rectangle that contains the position of all the agents. Note that the LP problem attempts to minimize the quantity $r = \max\{S_x, S_y\}$.

We first consider a very special case with three agents. Let us assume that minimal formation by three agents is a rectangle and not necessarily a square where $S_x = S_y$. Then, there are four way in which this can be occur. This is shown in Figure 3.

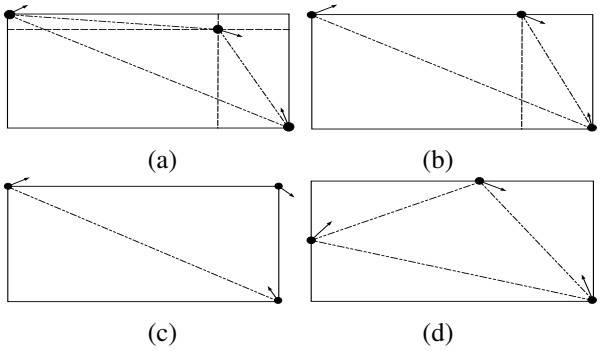


Fig. 3. Rectangle formation by three agents (a) Two agents at corner, one in the interior (b) Two agents at corner, one on edge (c) All the agents at corner (d) One agent at a corner and two agents on edges

Before we prove some general results on the formation of the agents after the LP is executed we will state a lemma that will be useful to prove the main results.

Lemma 1: If $C = \cos(\theta_i + \theta) - \cos(\theta_j + \theta)$ then there exists a θ such that $C < 0$ where $\theta_i, \theta_j \in (-\pi, \pi]$ and $\theta_i \neq \theta_j$.

Proof: Let, $\theta = (\pi - (\frac{\theta_i - \theta_j}{2}) - \Delta\phi)$. After replacing θ in $C = \cos(\theta_i + \theta) - \cos(\theta_j + \theta)$ we will get $C = -2\sin(\frac{\theta_i - \theta_j}{2})\sin(\Delta\phi)$. It is clear from the expression that we can make $C < 0$ by choosing $\Delta\phi$ properly. ■

Theorem 2: After executing the LP, square is the optimal formation for three agents.

Proof: We will prove this by contradiction. Suppose after executing the LP, the optimal formation is a rectangle. Let the position and orientation of the three agents be $x_{i0} = (p_{i01}, p_{i02}) \in \mathbb{R}^2$ and $\theta_{i0} \in (-\pi, \pi]$, respectively, for all $i \in \{1, 2, 3\}$. Let agents k and l be on the left edge and right edge of the rectangle. Without loss of generality, we can assume $S_{x0} > S_{y0}$ and $\Delta d < \frac{1}{4}\min\{(S_{x0} - S_{y0}), | p_{io1} - p_{jo1} |\}$, where $i, j \in \{1, 2, 3\}$. Let us define a broadcast control command $(\theta, \Delta d)$ such that $\Delta d > 0$ is very small. After broadcasting $(\Delta d, \theta)$, for some θ, the new positions will be

$$p_{i11} = p_{i01} + \Delta d\cos(\theta_{i0} + \theta)$$

$$p_{i12} = p_{i02} + \Delta d\sin(\theta_{i0} + \theta)$$

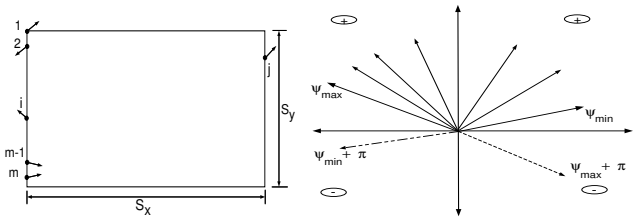
The new dimensions of the rectangle are given by S_{x1} and S_{y1}, where $S_{x1} = p_{k11} - p_{l11}$. As $S_{x0} > S_{y0}$ and $\Delta d < \frac{1}{4}\min\{(S_{x0} - S_{y0}), | p_{io1} - p_{jo1} |\}$ then $S_{y1} < S_{x1}$.

$$\begin{aligned} S_{x1} &= S_{x0} + \Delta d\{\cos(\theta_k + \theta) - cos(\theta_l + \theta)\} \\ &= S_{x0} + \Delta dC \end{aligned}$$

According to Lemma 1, there always exists θ such that $C < 0$. As $C < 0$ then $S_{x1} < S_{x0}$. This implies that there exist a broadcast command $(\Delta d, \theta)$ such that the maximum length of the sides of the rectangle can be further reduced. This implies that the optimal solution of the LP problem for three agents can not yield a rectangle. It has to be a square.

This proof is valid for Fig. 3(a), 3(b) and 3(d). The proof for Fig. 3(c) will be taken care of by the next Theorem. ■

Theorem 3: When the initial conditions are such that only one agent is on one edge of the longer span and m agents ($m > 1$) are on the other edge of the larger span, then the LP solution leads to a square formation.



(a) (b)

Fig. 4. An illustration for the proof of Theorem 3.

Proof: Consider Fig. 4 where $S_x > S_y$. The X distance between agent j and the other m number of agent $S_x = p_{j1} - p_{i1}$, where $i \in \{1, \dots m\}$. Let $\Delta d < \frac{1}{4}(S_x - S_y)$. After broadcasting $(\Delta d, \theta)$, the new dimensions of the rectangle are $S_x^{'}$ and $S_y^{'}$. Since $\Delta d < \frac{1}{4}(S_x - S_y)$, we have $S_y^{'} < S_x^{'}$. We can write

$$\begin{aligned} S_x^{'} &= p_{j1} - p_{i1} + \Delta d(\cos(\theta_j - \theta) - \cos(\theta_i - \theta)) \\ &= S_x + \Delta d R \sin(\Phi_i + \theta) \end{aligned} \tag{8}$$

where $\Phi_i = \left(\frac{\theta_j + \theta_i}{2}\right)$ and R is a positive quantity. For a particular θ_j we will get a set of angles $\{\Phi_i\}$ for m agents. Let $\Phi_{max} = \max\{\Phi_i\}$ and $\Phi_{min} = \min\{\Phi_i\}$. Note that θ_i is fixed here. Let θ_{j1} contribute to Φ_{max} and θ_{j2} contribute to Φ_{min}. Then $\Phi_{max} = \frac{\theta_{j1} + \theta_i}{2}$ and $\Phi_{min} = \frac{\theta_{j2} + \theta_i}{2}$. So $\Phi_{max} - \Phi_{min} = \frac{\theta_{j1} - \theta_{j2}}{2} \leq \pi$. We will get $S_x^{'} < S_x$, when $\sin(\Phi_i + \theta) < 0$. The set of angles $\{\Phi_i\}$ will be within a bounded envelope of $(0, \pi)$ as $\Phi_{max} - \Phi_{min} \in (0, \pi)$. Then there always exist an angle θ such that all the envelope will come in the lower two quadrants such that all $\sin(\Phi_i + \theta) < 0$. This implies that there $S_x^{'} < S_x$. Thus the LP solution cannot yield a rectangular formation since there will be another formation which can be achieved by broadcasting and which will have a smaller value of $\max\{S_x, S_y\}$. ■

However, there may not exist a solution when the number of agents on the opposite edges are more than two. As an example we can show that four agents with position and orientation of $((4.004, 3.9128), 0^o), ((4.015, 2.8264), 5^o)$,$((0.0024, 0.9302), -1^o)$ and $((0.0049, 1.9007), 0.7^o)$ will be move to $((5, 4), 5^o$, $((5, 3), 10^o)$, $((1, 1), 4^o)$, $((1, 2), 5.7^o)$. Although r decreases from 2.0064 to 2 but $S_x \neq S_y$. The reason behind this can be explained as follows:

Let agent j and k be on one edge and a set of agent $\{i\}$ on the opposite edge where $i \in \{1, \dots, m\}$. For agent j there will be a set of angles $\{\Phi_{ij}\}$ and for agent k there will be a set of angles $\{\Phi_{ik}\}$. For agent j, $\Phi_{max_j} = \max\{\Phi_{ij}\}$ and $\Phi_{min_j} = \min\{\Phi_{ij}\}$ and for agent k, $\Phi_{max_k} = \max\{\Phi_{ik}\}$ and $\Phi_{min_k} = \min\{\Phi_{ik}\}$. The range of $\Phi_{max_j} - \Phi_{min_j}$ and $\Phi_{max_k} - \Phi_{min_k}$ will both be $(0, \pi)$. Now, the range of $\max\{\{\Phi_{ij}\} \cup \{\Phi_{ij}\}\} - \min\{\{\Phi_{ij}\} \cup \{\Phi_{ij}\}\}$ will be greater than $(0, \pi)$. In which case there does not exist any common θ such that the X or Y axis distance will be reduced. They will remain at the same position after executing the LP.

Theorem 4: If the solution of the LP problem yields a square formation then the number of agents on the boundary of the square is more than two.

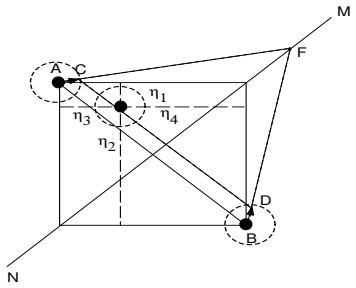


Fig. 5. An illustration for the proof of Theorem 4.

Proof: We will prove this by contradiction.Let, the number of agent on the square are two and they are located at diagonally opposite corners. The other agents are the interior of the square. For the sake of simplicity we will consider only three agents. This is given in Fig. 5,where MN is the Voronoi edge between agents located at A and B. According to Theorem 1, there always exists (θ, d) such that they can meet at an unique point on the Voronoi edge (MN). Let the unique meeting point be F. Let us define a very small positive quantity Δd such that $\Delta d < min\frac{1}{4}\{\eta_1, \eta_2, \eta_3, \eta_4\}$. After broadcasting $(\theta, \Delta d)$ the agents will move from their positions. The new position of agents A and B are C and D, respectively. The interior agent will remain in the interior of the square. CD is the new diagonal of the square. We can show that $\triangle ABF \sim \triangle CDF$ and so $CD \| AB$ and $CD < AB$. This implies that further improvement of the square is possible. This is a contradiction. ■

V. Achieving Perfect Consensus

The solution of the linear programming (LP) problem will yield control instructions which can be broadcast to all the agents. The agents will move to a new position or within a new square region of smaller area. It can be shown that no further improvement of the performance (reduction in r) can be achieved by repeated use of the algorithm. In other words, repeated application of the LP algorithm with the new final positions will not reduce the value of r any further.

Suppose we represent the LP algorithm as an operation L on the initial conditions that yields the solution as,

$$L(x_{i0}, \theta_{i0} | i = 1, \dots, n) = (u_1^*, u_2^*, r*, x_{if}, \theta_{if}) \tag{9}$$

then,

$$L(x_{if}, \theta_{if} | i = 1, \ldots, n) = (0, 0, r*, x_{if}, \theta_{if}) \quad (10)$$

That is, there will be no further change in the performance measure r. In other words, (x_{if}, θ_{if}) is a stationary point so far as the LP algorithm is concerned.

We can generalize this process by assuming that each step in the iteration is denoted by the index k, with the first step in the iteration as $k = 1$. We call this the *unperturbed case* as the solution of the LP is directly implemented by the agents without any perturbation to the solution.

Theorem 5: In the unperturbed case, for $k \geq 2$, $u^*_{1,k} = u^*_{2,k} = 0$ and $x_{i,k+1} = x_{i,k}$; $\theta_{i,k+1} = \theta_{i,k}$. This means that repeated use of the LP solution on subsequent positions will not reduce r.

Proof: We will prove this by contradiction. Suppose for a given initial condition $(x^1_{i0}, \theta^1_{i0})$ we have $r = r_0$ as the measure of proximity of the agents. Applying the LP algorithm we obtain u^{1*}_1, u^{1*}_2, r_1, and the final positions as $(x^1_{if}, \theta^1_{if})$. Now, considering the initial conditions as $(x^2_{i0}, \theta^2_{i0}) = (x^1_{if}, \theta^1_{if})$ and applying the LP algorithm let us assume that we get $u^{2*}_1 \neq 0, u^{2*}_2 \neq 0$, $r_2 < r_1$, and the final positions as $(x^2_{if}, \theta^2_{if})$. Then, let us define $\hat{u}_i = u^{1*}_i + u^{2*}_i$, $i = 1, 2$. It can be shown that if in the first step $\hat{u}_i$ is used it would yield a $r = r_2 < r_1$, which implies that r_1 was not a solution to the LP. This is a contradiction. ■

Now, consider a *perturbed case* where, the agents receive a broadcast command containing the LP solution and a command to randomly perturb the final orientation angle after the LP solution has been implemented. This process is shown in Figure 6, where, the orientation angle, after implementing the LP solution is perturbed by each agent as follows,

$$\hat{\theta}_{i,k+1} = \theta_{i,k+1} + \nu_{i,k+1} \quad (11)$$

where, the perturbation angle $\nu_{i,k+1}$ is given by $\nu_{i,k+1} = \eta_i \beta$. η_i is a random number generated by each agent independently and β is a scaling angle which is common to all the agents. The scaling angle can be set manually and η can be generated through various distributions. Here, we consider both normal distribution and uniform distribution.

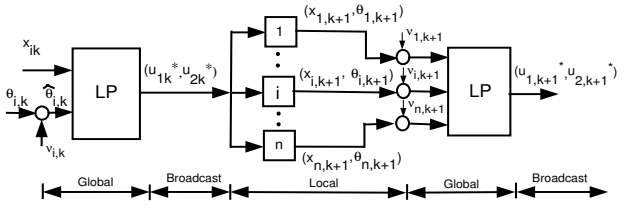


Fig. 6. The perturbed case

VI. Achieving Positional Consensus at Desired Point

In the previous section, we consider the problem of positional consensus, but did not have control over at which the agents can meet. Suppose we have a pre-specified meeting point then we can achieve this by slightly modifying the previous formulation. In this modified form we define the meeting point as the center of the agent formation and is denoted by (z_1, z_2). Now the input to the LP are the initial positions, initial orientation and the meeting point. We can formulate the modified linear programming problem as,

Minimize r

Subject to

$$-r \leq p_{i1} + u_1 \cos\theta_{i0} - u_2 \sin\theta_{i0} - z_1 \leq r \quad (12)$$

$$-r \leq p_{i2} + u_1 \sin\theta_{i0} + u_2 \cos\theta_{i0} - z_2 \leq r \quad (13)$$

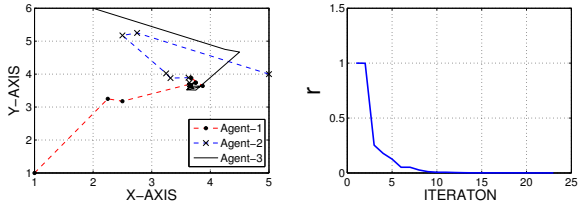
$$i = 1, \ldots, n.$$

$$r \geq 0 \quad (14)$$

The above is a linear programming problem with the decision vector as (r, u_1, u_2). Note that the number of decision variables has reduced over the previous formula. In the next section we will give simulations results.

VII. Simulation Results

In the first set of simulations we start with three agents. Initially we consider $((1, 1), 45^o)$, $((5, 4), 135^o)$, and $((2, 6), -45^o)$ as the initial position and orientation angle of the three agents. Using the perturbation technique, with normal distribution for η and a scaling angle of $\beta = 120^o$, the agents converge to a point after a few iterations (see Fig. 7). The variation in the x and y coordinates of the three agents against the number of iterations are also shown. The convergence criterion for terminating the simulation was when the value of r became less that 2×10^{-4}. We continue



(a) (b)

Fig. 7. Consensus with three agents (a) Trajectory of the agents (b) Reduction in r with iteration

with our study with three agents. In Fig. 8 we plot the average number of iterations needed, and the average length of path traveled by each agent, to achieve convergence, as a function of the scaling angle β for the two cases when the random number η is generated by a uniform distribution or by a normal distribution. These results are given by averaging over 200 trials. We also plot the standard deviations. From these results we can conclude that using larger scaling angle reduces r faster than when the scaling angles are small. Also, using normal distribution gives better results than uniform distribution.

Next, we consider larger number of agents (10 and 15). We use normal distribution and scaling angle of 180^o for the perturbation. The convergence criterion is also relaxed

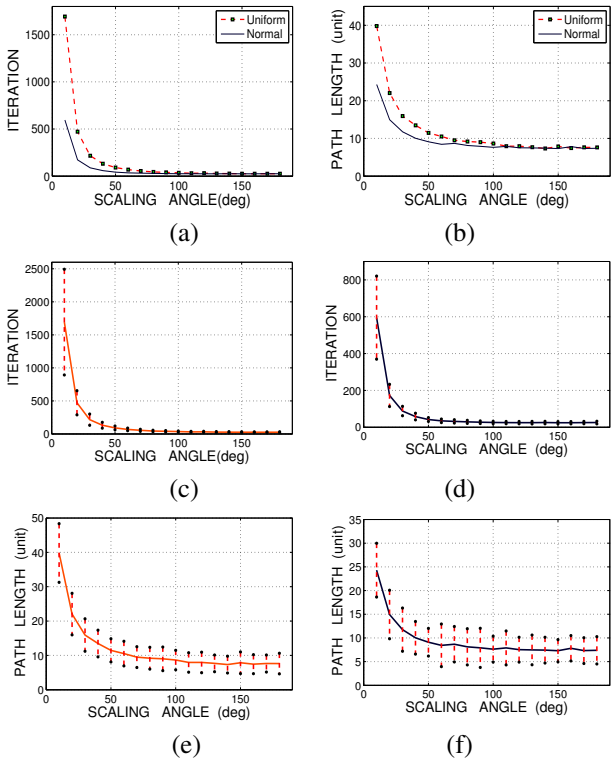


Fig. 8. Convergence results (a) Average number of iterations (b) Average length of path (c) Standard deviations for number of iterations: Uniform distribution (d) Standard deviations for number of iterations: Normal distribution (e) Standard deviations for path length: Uniform distribution (f) Standard deviations for path length: Normal distribution

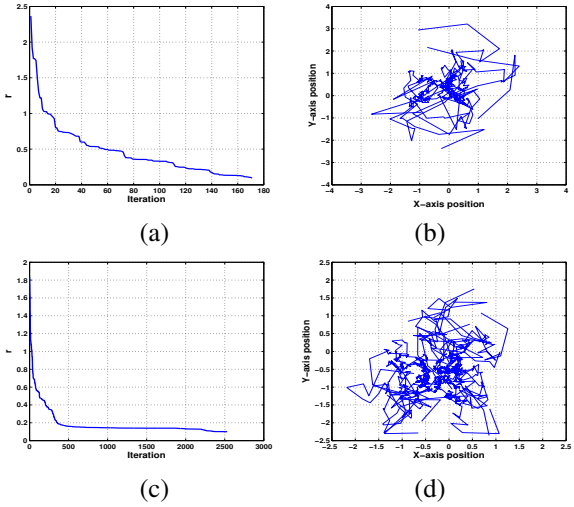


Fig. 9. Convergence results (a) r vs number of iterations for 10 agents (b) Trajectory for 10 agents (c) r vs number of iterations for 15 agents (d) Trajectory for 15 agents

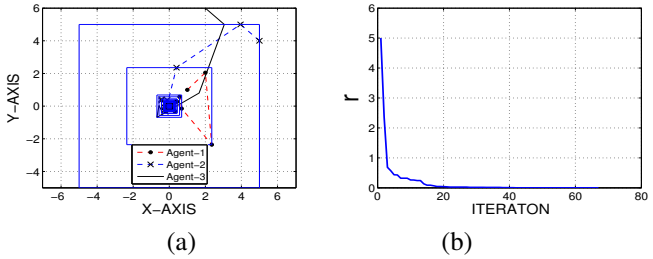


Fig. 10. Consensus with three agents at specified point (a) Trajectory of the agents (b) Reduction in r with iteration

to $r \leq 0.1$. The results are shown in Figure 9. These results show that the computation time and the number of iterations rises with the number of agents. This is expected since the complexity of the algorithm is same as that of the LP algorithm.

To demonstrate the result that the agents can be meet any pre-specified point, we consider same set of initial position and orientation angle($((1,1), 45^o)$, $((5,4), 135^o)$, and $((2,6), -45^o)$) of the three agents. The meeting point we set as origin $(0,0)$. The result is illustrated in Fig. 10.

VIII. Conclusions

In this paper we considered the problem of controlling a group of agents to converge at a location using only broadcast control input (identical control) for all the agents. The results shows that it is possible for a group of agents to meet at a location by sending them a sequence of simple and exactly identical instruction. The location point can be pre-specified. We were able to show that introducing a perturbation in the broadcast command helped to achieve point convergence which was not possible earlier. We also proposed a novel linear programming based solution approach that is computationally less intensive than the SOCP technique proposed in the literature. There are several opportunities for future work in this direction. It seems possible to extend this work to consider process noise, sensor uncertainty, and the presence of obstacles in the environment. Moreover, the algorithm can most probably shown to be robust to failures in terms of packet loss or failure of agents.

References

[1] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen and O. Schochet, Novel type of phase transistions in a system of self-driven particles , *Phys. Rev. Lett.*, vol. 75, 1995, pp 1226-1229.

[2] G. Antonelli, and S. Chiaverini, Kinematic control of platoons of autonomous vehicles, *IEEE Trans. Robotics and Automation*, vol. 22(6), 2006, pp 1285-1292.

[3] B. V. Gervasi and G. Prencipe, Coordination without communication: The case of the flocking problem, *Discrete Applied Mathematics*, vol. 143, 2004, pp 203-223.

[4] M. Cieliebak, P. Flocchini, G. Prencipe and N. Santoro, Solving the robots gathering problem , *In Proc. 30th Int. Colloq. on Automata, Language and Programming*, 2003, pp 1181-1196.

[5] I. Suzuki and M. Yamashita, Distributed anonymous mobile robots , *SIAM J. on Computing* , vol. 28, 1999, pp 1347-1363.

[6] T. Bretl, "Control of Many Agents Using Few Instructions", *in Third Robotics Science and System Conference* , Atlanta, 2007.

[7] B.R. Donald, C.G. Levey, C.D. Mcgray, I. Paprotny and D. Rus, An untethered,electrostatic, globally controllable MEMS micro-robot, *J. Microelectromech. Syst*, vol. 15(1), 2006, pp 1-15.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[9] H. Ando, Y. Oasa and I. Suzuki, Distributed Memoryless Point Convergence Algorithm for Mobule Robots with Limited Visibility, *IEEE Tran. of Robotics and Automation*, vol. 15(5), 1999, pp 818-828.

[10] W. Rudin, *Principles of Mathematicla Analysis*, McGRAW-HILL Book Company, 1976.

Randomization Mechanisms Based Positional Consensus in Homogeneous Multi-Agent Systems

Kaushik Das[a,b], Debasish Ghose[a]

[a]Guidance, Control, and Decision Systems Laboratory, Department of Aerospace Engineering, Indian Institute of Science, Bangalore
[b]Corresponding Author

Abstract: In this paper we present a modified broadcast control algorithm for controlling a group of homogeneous agents to achieve positional consensus. We make an assumption that the agents are capable of generating a flag which indicates whether agent will be active or passive. This proposed technique is based on the constraint that all agents must be given the same control input through a broadcast communication mechanism. The control command is computed using state information of active agents or active and passive agents in a global framework. The control input is implemented by the active agents in a local coordinate frame. We propose some modified broadcast control mechanisms based on linear programming with or without introducing random perturbation in the orientation of the active agents. We show that even without introducing perturbation in the orientation angle positional consensus is possible if we adopt random flag setting. We make a comparison between several algorithm that have been designed as combinations of the two randomization mechanism. Moreover, we extend the method to achieve positional consensus through this modified algorithm at a pre-specified point. The effectiveness of the approach is illustrated through simulation results.

Key words: Multi-agent systems, Linear Programming, Broadcast Communications

1. INTRODUCTION

The principle of using multiple agents is motivated by the idea that instead of using a highly sophisticated and expensive agent, it may be advantageous in certain situations to use a group of small, simple, and relatively cheap agents. The group of agents can be used to accomplish various tasks in different environment such as tactical operations, exploratory space missions, remote monitoring with mobile sensor networks, avoidance of collision and over-crowding in automated air traffic control, cleanups of toxic spills, fire fighting and cooperative search with unmanned air vehicles.

One of the problems that is of paramount importance in multi-agent systems is that of achieving consensus, that is, achieving identical values for some specified subset of the states of the agents. For instance, the agents may try to converge to the same direction of movement [1] or they might want to converge to a point. Both are problems in achieving consensus. If we have a centralized system with perfect information, then achieving consensus is a trivial matter, since the central controller can instruct each agent suitably to reach a common consensus point. However, if the communication system has a constraint on the number of messages that it can communicate, then one may opt for a broadcast protocol where the central controller will communicate simple and identical instructions to all the agents through a broadcast mechanism. This is the framework in which we address the problem of consensus in this paper. Other papers that address similar problem are [2, 3, 4, 5].

This problem we address in the paper was motivated by a recent paper by Bretl [6] where a control strategy for a group of micro-robots is developed to perform a useful task even when every robot receives the same control signal. The paper considers point robots with simple kinematics. It was shown that when there are only two agents, there exists a broadcast control command (that is, both agents receive identical instructions from the central controller) using which both agents can meet at the same location at the same time, for almost all initial conditions. However, if the number of agents is more than two, then the best that the agents can achieve is to come close to each other within a certain distance (measured by the radius of the smallest disc that contains all the agents positions), which is a function of the initial conditions. Bretl [6] formulates an optimization problem that minimizes the radius of the disc, and proposes a solution using the sec-

Email addresses: kaushikdas@aero.iisc.ernet.in (Kaushik Das), dghose@aero.iisc.ernet.in (Debasish Ghose)

[1]This project is partially supported by AOARD/AFOSR.

ond order cone programming (SOCP) technique [7]. However, using this strategy the agents cannot be made to converge to a point. Once the solution of the SOCP is implemented, no further improvement is possible.

Bretl's paper was in turn motivated by an interesting paper by Donald et al. [8] on the development of untethered, steerable micro-robots, where every robot receives the same power and control signal through an underlying electrical grid. These robots mainly have two parts. One is an untethered scratch drive actuator used mainly for forward locomotion. The other part is a cantilevered steering arm used for turning through frictional contact with the substrate. These micro-robots are simple in construction and can operate only in a local coordinate frame. They do not have sensory capabilities to determine their position and orientation in the global coordinate frame. Neither do they have capabilities to localize themselves in relation to their neighboring objects or other robots.

Based on the above paradigm Das and Ghose [9] proposed a strategy that uses the basic Bretl's model with an additional randomization feature that introduced a random perturbation on the angular orientation of the agents to achieve positional consensus or point convergence on repeated application of the algorithm without compromising the broadcast constraint on the control command. This method has extended to the case where the agents can be made to converge to any pre-specified point. The paper [9] also proposes an optimization problem for this task that is based on a linear programming formulation instead of SOCP technique and shows improved computational performance. However, the proposed LP based method with the randomization feature had a serious drawback when the number of agents is very large. This was observed in simulations carried out in [9]. In the present paper we introduce another randomization feature that not only introduces a random perturbation to the agents' angle but also sets a flag that randomly assigns active or passive role to the agents, where an active agent implements the broadcast control while the passive agents ignore it. We show that this additional features helps in achieving consensus or point convergence of even large number of agents.

2. FORMULATION AND SOLUTION FOR TWO AGENTS

We will first pose the problem in a general framework and then address the two agents case to clarify many of the assumptions and concepts discussed in the previous section.

Assume that n agents are located on an obstacle-free plane. We assume that the central controller has access to the global state of the system which, in this case, consists of the position ($x_i \in \mathbb{R}^2$) and orientation ($\theta_i \in (-\pi, \pi]$) of the agents, $i = 1, \ldots, n$. The central controller computes a common local control for the agents and broadcasts it to the agents for implementation. The local control is in the form of a tuple (θ, d), which is interpreted by each agent in its local frame of reference. Here, θ refers to the angle by which each agent changes its orientation, and d is a scalar that refers to the distance by which each agent moves after effecting the orientation change. Note that due to the broadcast mechanism (θ, d) is the same for all the agents. Also, the local frame of reference for each agent is centered at the agent's location and its reference axis is oriented along its current orientation. As an illustration see Fig. 1, where agents are shown located initially at x_{i0} with initial orientation θ_{i0} in the global reference frame.

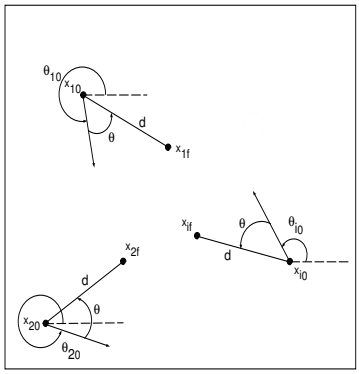


Figure 1: Basic configuration

If the control command broadcast to all the agents is (θ, d), then the agents implement it in their local coordinate frame by each of them changing their orientation by the same angle θ and advancing by the same distance d to reach the final destination x_{if}. Even in this figure it can be seen that by doing this the agents have come closer to each other. Our objective is to determine a (θ, d) such that the agents can achieve the closest proximity with each other.

Theorem 1. *For two agents, for all initial conditions of the agents except when $\theta_{10} = \theta_{20}$, there exists a control (θ, d) using which point convergence can be achieved.*

Proof. From Fig. 1, assuming only two agents, we have,

$$x_{if} = x_{i0} + R(\theta_{i0})R(\theta)\begin{bmatrix} d \\ 0 \end{bmatrix}, \quad i = 1, 2 \tag{1}$$

where, $R(\alpha)$ is the rotation matrix,

$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{2}$$

Letting $x_{1f} = x_{2f}$ we obtain,

$$R(\theta)\begin{bmatrix} d \\ 0 \end{bmatrix} = [R(\theta_{20}) - R(\theta_{10})]^{-1}(x_{10} - x_{20}) \tag{3}$$

The above equation can be solved easily for d and θ if the inverse on the right hand side exists, which happens when $\theta_{10} \neq \theta_{20}$. □

The above theorem shows that it is possible to use a broadcast control command to make two agents meet at the same location simultaneously for almost all initial conditions. However, the solution is also unique and hence the location of the meeting point cannot be chosen arbitrarily.

It can be seen that when the number of agents is more than two, each pair gives rise to a different unique meeting point. Thus, there does not exist a common control command to be broadcast so that all the agents meet at a point. In the absence of such a command, the best that can be done is to determine a (θ, d) which brings the agents in closest proximity with each other. Note that in this case (θ, d) may not be unique.

In the next two sections we will propose a linear programming based solution and a randomized mechanism to obtain consensus without compromising the broadcast based control mechanism.

3. A LINEAR PROGRAMMING SOLUTION FOR MULTIPLE AGENTS

Let the initial position and initial orientation of the n agents be $x_{i0} = (p_{i1}\ p_{i2}) \in \mathbb{R}^2$ and $\theta_i \in (-\pi, \pi]$, respectively, for all $i \in \{1, ..., n\}$. As before, we define the control command to be broadcast as (θ, d). We define our performance measure as the half length, denoted by $r > 0$, of the side of a square oriented along the global coordinate frame, and containing all the final positions of the agents. Let this square be centered at $z = (z_1, z_2) \in \mathbb{R}^2$.

Assuming that all the agents execute the command (θ, d), their final positions, given by $x_{if} = [q_{i1}\ \ q_{i2}] \in \mathbb{R}^2$ will be,

$$x_{if} = x_{i0} + R(\theta_{i0})R(\theta)\begin{bmatrix} d \\ 0 \end{bmatrix} \tag{4}$$

That is,

$$\begin{bmatrix} q_{i1} \\ q_{i2} \end{bmatrix} = \begin{bmatrix} p_{i1} \\ p_{i2} \end{bmatrix} + \begin{bmatrix} \cos\theta_{i0} & -\sin\theta_{i0} \\ \sin\theta_{i0} & \cos\theta_{i0} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{5}$$

where, $u_1 = d\cos\theta$ and $u_2 = d\sin\theta$ are the control variables that replace (θ, d). Note that Eqn. (5) are linear equations. Now, we formulate the linear programming problem as,

Minimize r

Subject to

$$-r \le p_{i1} + u_1\cos\theta_{i0} - u_2\sin\theta_{i0} - z_1 \le r \tag{6}$$

$$-r \le p_{i2} + u_1\sin\theta_{i0} + u_2\cos\theta_{i0} - z_2 \le r \tag{7}$$

$$i = 1, \ldots, n.$$

$$r \ge 0 \tag{8}$$

The above is a linear programming problem with the decision vector as (r, z_1, z_2, u_1, u_2). Note that the decision vector remains the same irrespective of the number of agents. Only the number of inequality constraint increases with the number of agents. Also, note that z_1,z_2,u_1 and u_2 are free variables and can take both positive or negative values. An illustration of this process is shown in Figure 2.

The solution of the linear programming (LP) problem will yield control instructions which can be broadcast to all the agents using which the agents will move to a new position or within a new square region of smaller area, that is, a square

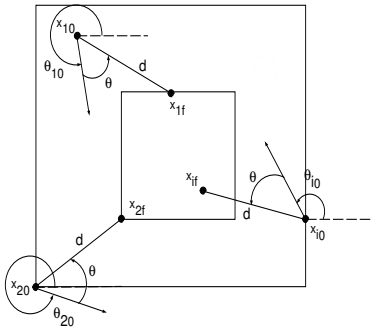


Figure 2: An illustration of how the square reduces in size using the LP solution

with a reduced side length. It can be shown that no further improvement of the performance (that is, reduction in r) can be achieved by repeated use of the algorithm.

Suppose we represent the LP algorithm as an operation L on the initial conditions that yields the solution as,

$$L(x_{i0}, \theta_{i0} | i = 1, \ldots, n) = (u_1^*, u_2^*, r*, x_{if}, \theta_{if}) \tag{9}$$

then,

$$L(x_{if}, \theta_{if} | i = 1, \ldots, n) = (0, 0, r*, x_{if}, \theta_{if}) \tag{10}$$

That is, there will be no further change in the performance measure r. In other words, (x_{if}, θ_{if}) is a stationary point so far as the LP algorithm is concerned.

We can generalize this process by assuming that each step in the iteration is denoted by the index k, with the first step in the iteration as $k = 1$. Then we can represent the process as in Figure 3. We call this the *unperturbed case* as the solution of the LP is directly implemented by the agents without any perturbation to the solution. The following result has been proved in [9] and is stated here for completion.

Theorem 2. *In the unperturbed case, for $k \ge 2$, $u_{1,k}^* = u_{2,k}^* = 0$ and $x_{i,k+1} = x_{i,k}$; $\theta_{i,k+1} = \theta_{i,k}$. This means that repeated use of the LP solution on subsequent positions will not reduce r.*

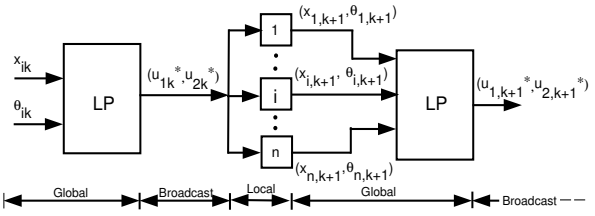


Figure 3: Generalization of the unperturbed case

4. RANDOMIZATION MECHANISMS

Now, consider a *perturbed case* where the agents use a randomization mechanism to modify their states and implement broadcast command containing the LP solution. The

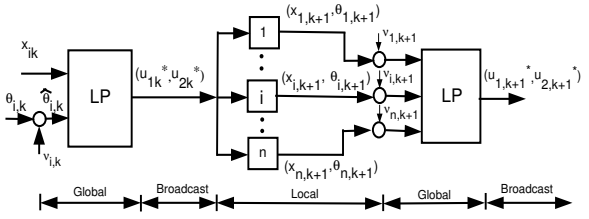


Figure 4: Perturbation in angle

randomization mechanism contains two components, in one of which a random perturbation is introduced in the orientation angle of the agent and in the other a flag is set by which an agent designates itself as active or passive. In the angle perturbation case, the final orientation of each agent is perturbed and is used by the central controller to implement the LP algorithm. This process is shown in Figure 4, where, the orientation angle, after implementing the LP solution is perturbed by each agent as follows,

$$\hat{\theta}_{i,k+1} = \theta_{i,k+1} + \nu_{i,k+1} \tag{11}$$

where, the perturbation angle $\nu_{i,k+1}$ is given by,

$$\nu_{i,k+1} = \eta_i \beta \tag{12}$$

where, η_i is a random number generated by each agent independently and β is a scaling angle which is common to all the agents. The scaling angle can be set manually and η can be generated through various distributions. In Eqn. (11) $\nu_{i,k+1}$ represents the perturbation angle. The perturbation angle for different agents are different. This kind of random mechanism has been studied in [9] and has been found to be deficient when the number of agents is large, as they do not come closer than a certain value even after a large number of iterations.

The main idea behind the flag setting approach is that each agent (i) randomly generates a flag f_i, that can take one of two values. One is high (f_h), denoting an active agent and other is low (f_l), denoting a passive agent. The set of agents that have high flags (active) is denoted by R_a and the set of agents that have low flags (passive) is R_p.

Let the set of active agents at time t be $R_a(t) = \{i \mid f_i = f_h\}$ and the set of passive agents at time t be $R_p(t) = \{i \mid f_i = f_l\}$ where $i \in \{1, \ldots, n\}$. An active agent is one that will implement the broadcast command and change its state whereas a passive agent will ignore the command and remain stationary. The flag value for each agent may or may not be observed by the central controller (CC). The central controller have information of position and orientation of all the agents (active and passive) at each step. The control command (u_1^*, u_2^*) computed by the central controller will be broadcast to all the agents, but only the active agents will move.

Let the position-orientation of active and passive agents be $(x_{a,k}, \theta_{a,k})$ and $(x_{p,k}, \theta_{p,k})$, respectively. After receiving the broadcast command the new position of the active and passive agents are

$$x_{a,k+1} = x_{a,k} + R(\theta_{a,k})R(\theta)\begin{bmatrix} d \\ 0 \end{bmatrix} \tag{13}$$

$$x_{p,k+1} = x_{p,k} \tag{14}$$

where (θ, d) is the broadcast control command.

5. ALGORITHMS

In this section we will propose five different algorithms obtained as combination of the two randomization mechanisms described in the previous section. They are denoted as A, (A, F_O), (A, F_N), F_O, F_N and are shown in Fig. 5. The algorithm A that introduced perturbation only in the angle has been studied in [9].

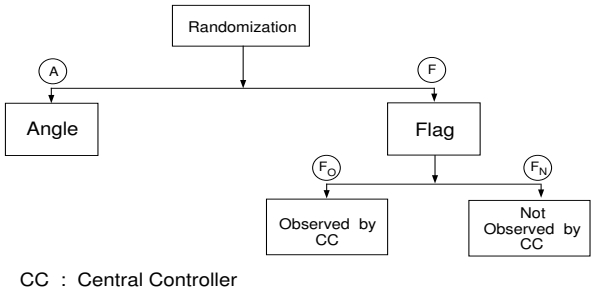


Figure 5: Taxonomy of the randomization mechanism based algorithm

5.1. Algorithm (A, F_O)

In this case, the active agents in $R_a(t)$ whose flag is high will perturb its orientation randomly. Then, the central controller will observe the positions, orientations and flags of all agents, but will compute the control command (θ, d) using the position and orientation of the active agents only. The control command will be broadcast to all the agents. After receiving the control command only the active agents will change their state. The passive agents will remain stationary. This implies that,

$$L((x_{a,k}, \widehat{\theta}_{a,k})) = (u_1^*, u_2^*) \tag{15}$$

where, $\widehat{\theta}_{a,k} = \theta_{a,k} + \nu_{a,k}$.

5.2. Algorithm (A, F_N)

In this case also the flag will be set randomly, as well as the orientation. However, the central controller does not have access to the flag information and will compute the control command (θ, d) using positions and orientations of both the active agents and passive agents. The control command will be broadcast and all the active agents will move from their respective positions. This implies that,

$$L((x_{a,k}, \widehat{\theta}_{a,k}), (x_{p,k}, \theta_{p,k})) = (u_1^*, u_2^*) \tag{16}$$

where, $\widehat{\theta}_{a,k} = \theta_{a,k} + \nu_{a,k}$.

5.3. Algorithm (F_O)

In this case the flags of the agents will be set randomly but the orientation will not be perturbed. The central controller will observe the flags, positions and orientations of all the agents and compute the control command for the active agents only. The central controller will broadcast control command and all active agents will move from their respective positions. This implies that,

$$L(x_{a,k}, \theta_{a,k})) = (u_1^*, u_2^*) \tag{17}$$

Note that in this case there is no perturbation in the agent orientation.

5.4. Algorithm (F_N)

In this case also the flag of the agents will be set randomly, but orientation will not be perturbed. The central controller will observe positions and orientations of all the agents and compute the control command based on the position and orientation of all agents. The central controller will broadcast the control command to all the agents but only the active agents will implement these and will move.

$$L((x_{a,k}, \theta_{a,k}), (x_{p,k}, \theta_{p,k})) = (u_1^*, u_2^*) \tag{18}$$

We put all the above algorithms in a tabular form in Table 1.

Table 1: The different performance for the algorithm (Y = Yes ; N = No)

Algorithm	Random Perturbation in Orientation	Random Flag Setting	LP Solved Using Agents	Movements of Agents
A	Y	N	all	all
(A, F_O)	Y	Y	active	active
(A, F_N)	Y	Y	all	active
F_O	N	Y	active	active
F_N	N	Y	all	active

6. ACHIEVING POSITIONAL CONSENSUS AT A DESIRED POINT

In the previous section, we considered the problem of positional consensus, but did not have control over the point at which the agents can meet. In [9] a scheme has been proposed to address this. We can define the meeting point (z_1, z_2) of the agent formation. In this case, the input to the LP are the initial positions, initial orientation and the meeting point. The linear programming formulation is the same as [9]. We can formulate the modified linear programming problem as,

Minimize r

Subject to

$$-r \leq p_{a1} + u_1 \cos\theta_{a0} - u_2 \sin\theta_{a0} - z_1 \leq r \tag{19}$$

$$-r \leq p_{a2} + u_1 \sin\theta_{a0} + u_2 \cos\theta_{a0} - z_2 \leq r \tag{20}$$

$$a = \{i \mid i \in R_a\}$$

$$r \geq 0 \tag{21}$$

The above is a linear programming problem with the decision vector as (r, u_1, u_2). Note that the number of decision variables has reduced over the previous formulation.

7. SIMULATION RESULTS

In the first set of simulations consider three agents. Initially we consider $((1,1), 45^o)$, $((5,4), 135^o)$, and $((2,6), -45^o)$ as the initial position and orientation angle of the three agents. Using algorithm A all three agents converge to a point after a few iterations (see Fig. 6). The variation in the x and y coordinates of the three agents against the number of iterations are also shown. The convergence criterion for terminating the simulation is gives as value of r becoming less that 2×10^{-4}.

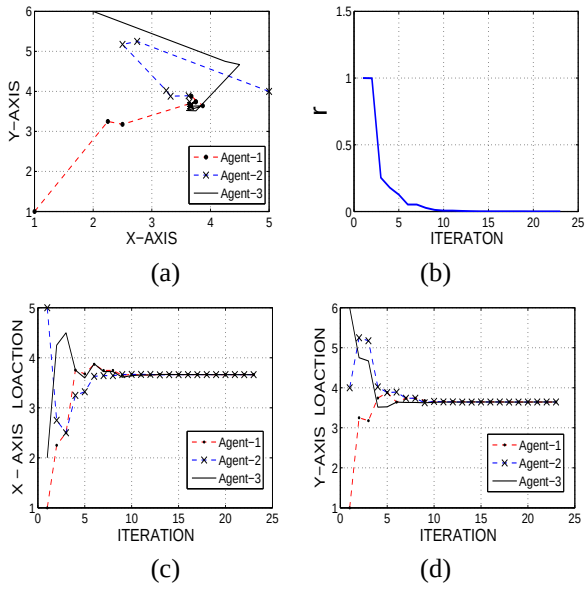


Figure 6: Consensus with three agents using algorithm A (a) Trajectory of the agents (b) Reduction in r with iteration (c) Convergence in the X-coordinate (d) Convergence in the Y-coordinate

Next, we consider larger number of agents (5, 10, 15 and 20). The convergence criterion is also relaxed to $r \leq 0.1$. We can see after some iteration the rate of decrement of performance radius r is very small. For agents 10, 15, 20 the performance radius is not improving after some iterations. The results are shown in Figure 7. These results show that the computation time and the number of iterations rise with the number of agents. This is expected since the complexity of the algorithm is the same as that of the LP algorithm.

Next we consider Algorithm (A, F_O). Note that in this case the control command is computed by considering only the active agents. We consider 5 agents in this simulation. The simulation result is given in Fig. 8. The convergence criterion for terminating the simulation is gives as value of r becoming less than 0.1. The simulation result shows that the agent are not converging to a point. In fact they seem to be sometimes going apart from each other and sometimes coming closer.

The reason behind this is the following. The central controller is computing the control command based on the active agents information only and it is independent of passive agent information. This implies that the active agents move to a point defined by the LP solution without taking into consideration the presence of the passive agents. This leads to the active agents moving away from the passive agents. So after repeatedly using this algorithm the agent will not converge to a point.

Next we consider Algorithm (A, F_N). The central controller does not observe the flags and considers all the agents while compute the control command. Here we consider 5, 10, 15 and 20 agents for illustration. The convergence criterion for terminating the simulation is the same as (A, F_O). The simulation result is given in Fig. 9. We can see that this algorithm is able to bring a higher number of agents at a point. This algorithm are opposite to previous algorithm. The reason why this algorithm works better than (A, F_O) is that consideration of the passive agents in the LP computation keeps the agents swarm together. Thus, not have access to the flag information (by the central controller) actually helps in better convergence.

Next we consider the Algorithm F_O. Here the central controller observes the flag values and there is no perturbation in orientation. Here also we take 5 agents to demonstrate the result. The termination condition for simulations is the same. The simulation result is given in Fig. 10. We can see the agents do not converge to a point. The reason for not converging is same as Algorithm (A, F_O).

Next we consider the Algorithm F_N. Here the central controller does not observes the flags and there is no perturbation of angles. we consider 5, 10, 15 and 20 agents to demonstrate the result. The simulation result are given in Fig. 11. We can see the agents converge to a point for upto 15 agents. The simulation result (Fig. 11) shows that, this algorithm does not work for agent 20 agents but still it gives better result than algorithm A.

We see all the algorithms are not efficient for making the agents converge to a point. Table 2 shows performance comparison of all the algorithms. Table 3, Table 4 and Table 5 gives comparison of the number of iterations and of the computations time between the algorithms for 10 and 15 and 20 agents respectively. Note that NC denotes these cases where the algorithm did not make the agents converge to a point.

To demonstrate the result that the agents can meet at any pre-specified point, we consider 5 agents. Here the meeting point is $(0, 0)$. The result is given in Fig. 12. It shows that where the final point is specified all the algorithms will work. Table 6, Table 7 and Table 8 gives comparison of the number of iterations and of the computations time between the algorithms for 10 and 15 and 20 agents respectively when the meeting point is specified. The results shows that even the number of agents are higher, the algorithms (A, F_O) and F_O are taking few iteration to converge.We consider 50 agents also for the algorithms (A, F_O), F_O and the simulation result is shown in Fig. 13.

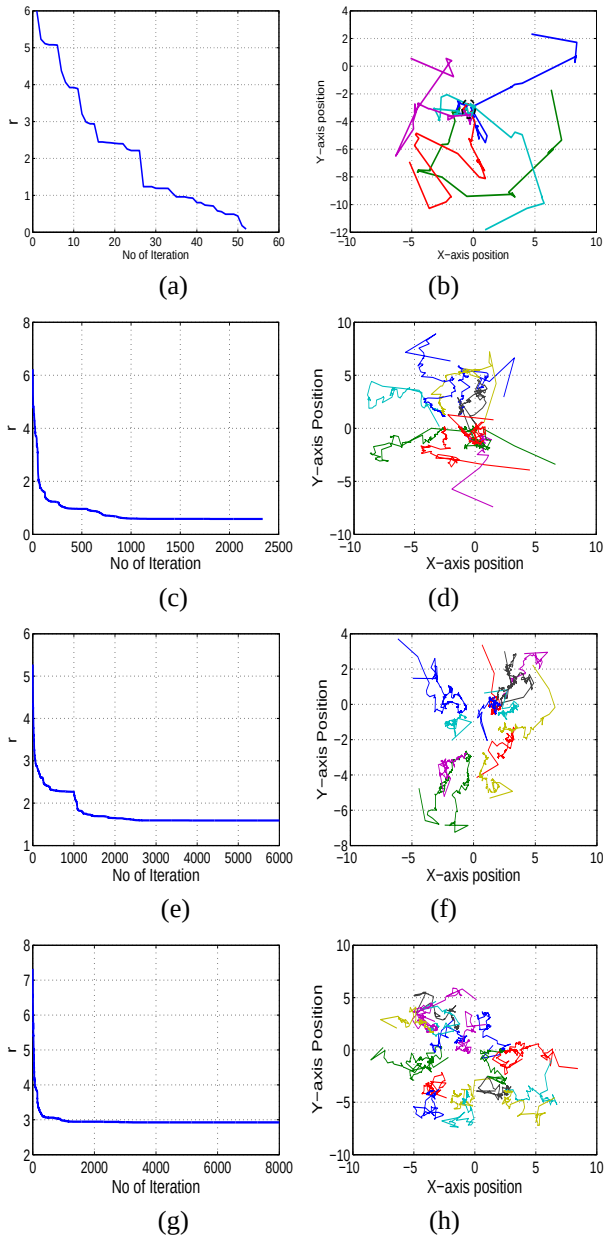


Figure 7: Algorithm A (a) r vs number of iterations for 5 agents (b) Trajectory for 5 agents (c) r vs number of iterations for 10 agents (d) Trajectory for 10 agents (e) r vs number of iterations for 15 agents (f) Trajectory for 15 agents (g) r vs number of iterations for 20 agents (h) Trajectory for 20 agents

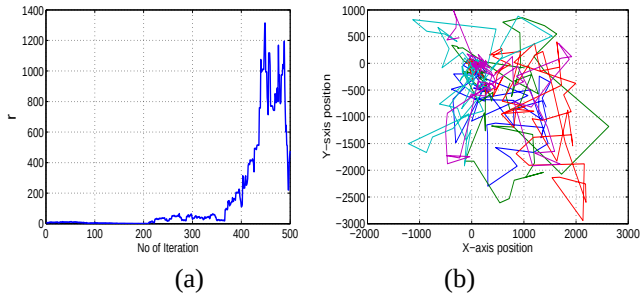
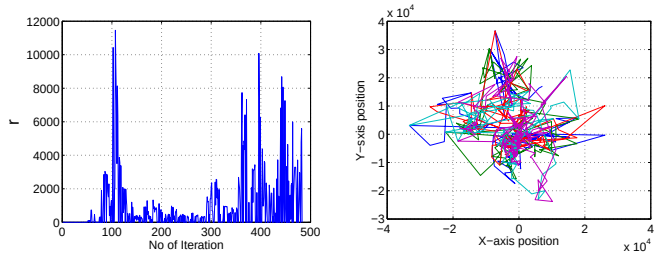


Figure 8: Algorithm (A, F_O) (a) Change in r with iteration (b) Trajectory of the agents

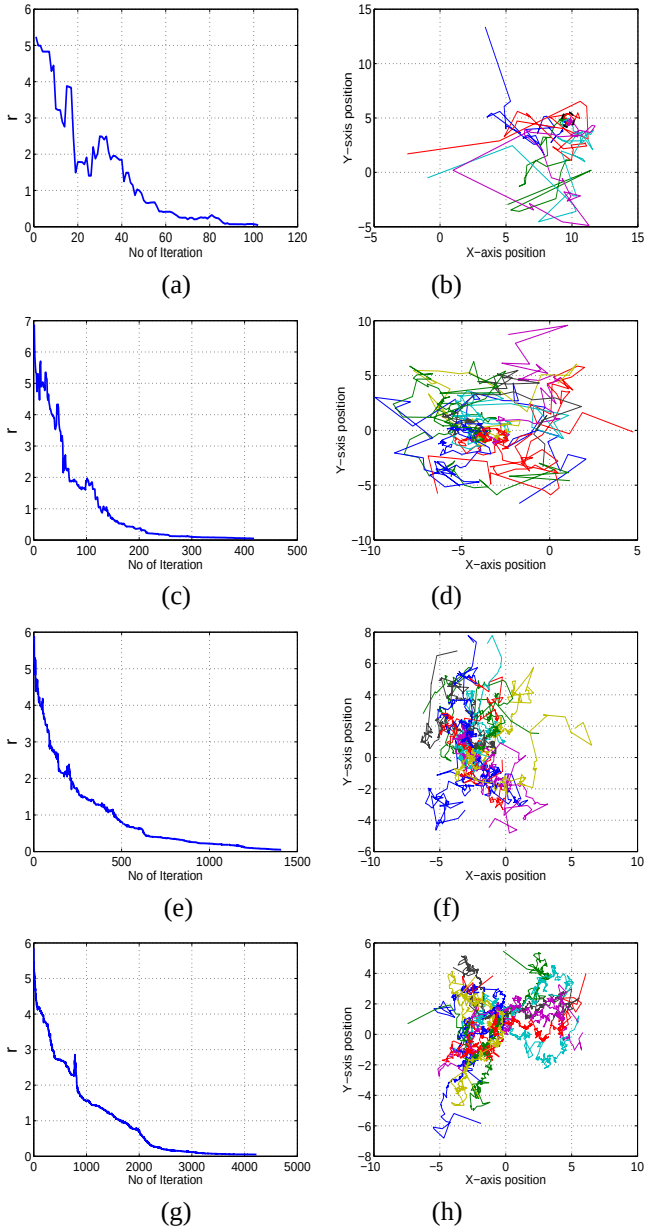
Table 2: Comparison between the algorithms for different number of agents

	Converge			
Algorithm	5	10	15	20
A	Y	Y	N	N
(A, F_O)	N	N	N	N
(A, F_N)	Y	Y	Y	Y
F_O	N	N	N	N
F_N	Y	Y	Y	N



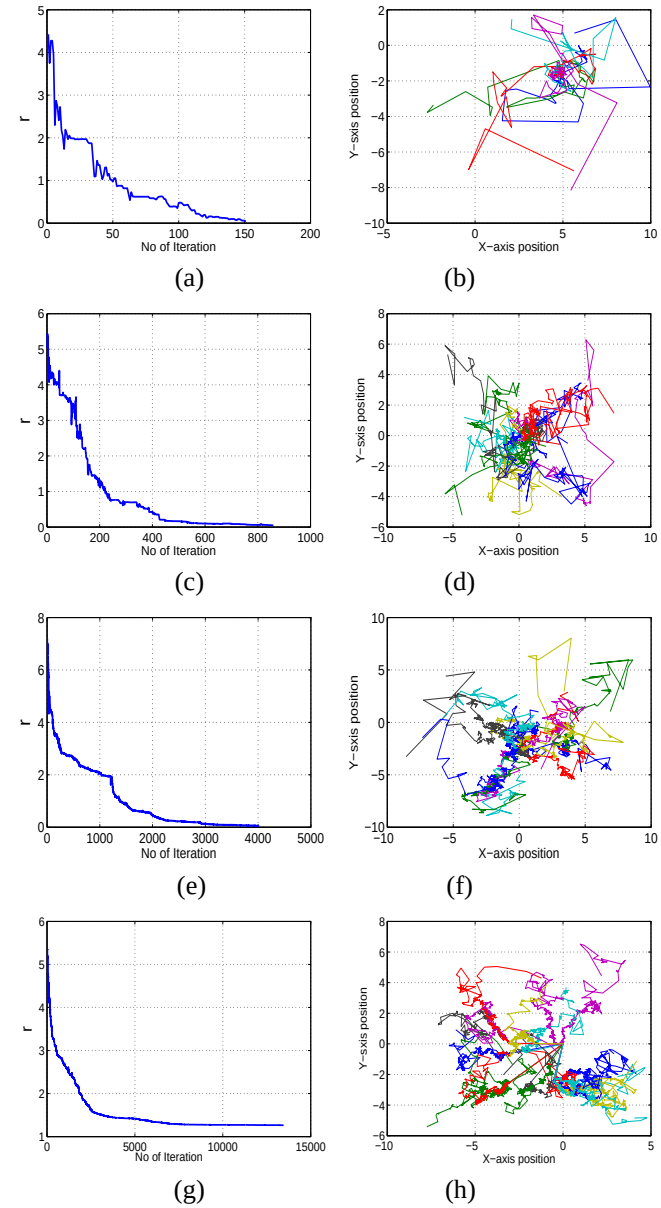
(a) (b)

Figure 10: Algorithm F_O (a) Reduction in r with iteration (b) Trajectory of the agents



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 9: Algorithm (A, F_N) (a) r vs number of iterations for 5 agents (b) Trajectory for 5 agents (c) r vs number of iterations for 10 agents (d) Trajectory for 10 agents (e) r vs number of iterations for 15 agents (f) Trajectory for 15 agents (g) r vs number of iterations for 20 agents (h) Trajectory for 20 agents



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 11: Algorithm F_N (a) r vs number of iterations for 5 agents (b) Trajectory for 5 agents (c) r vs number of iterations for 10 agents (d) Trajectory for 10 agents (e) r vs number of iterations for 15 agents (f) Trajectory for 15 agents (g) r vs number of iterations for 20 agents (h) Trajectory for 20 agents

Table 3: Comparison between algorithm for 10 agents, (NC : No Convergence)

	10 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	NC	NC
(A, F_N)	1262	26
F_O	NC	NC
F_N	1381	29

Table 4: Comparison between algorithm for 15 agents (NC : No Convergence)

	15 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	NC	NC
(A, F_N)	1874	59
F_O	NC	NC
F_N	5584	234

Table 5: Comparison between algorithm for 20 agents (NC : No Convergence)

	20 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	NC	NC
(A, F_N)	7949	1056
F_O	NC	NC
F_N	NC	NC

Table 6: Comparison between algorithm for 10 agents with specified point (NC : No Convergence)

	10 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	119	9
(A, F_N)	2193	135
F_O	118.25	8
F_N	4101	229

Table 7: Comparison between algorithm for 15 agents with specified point (NC : No Convergence)

	15 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	235	10
(A, F_N)	NC	NC
F_O	278	14
F_N	NC	NC

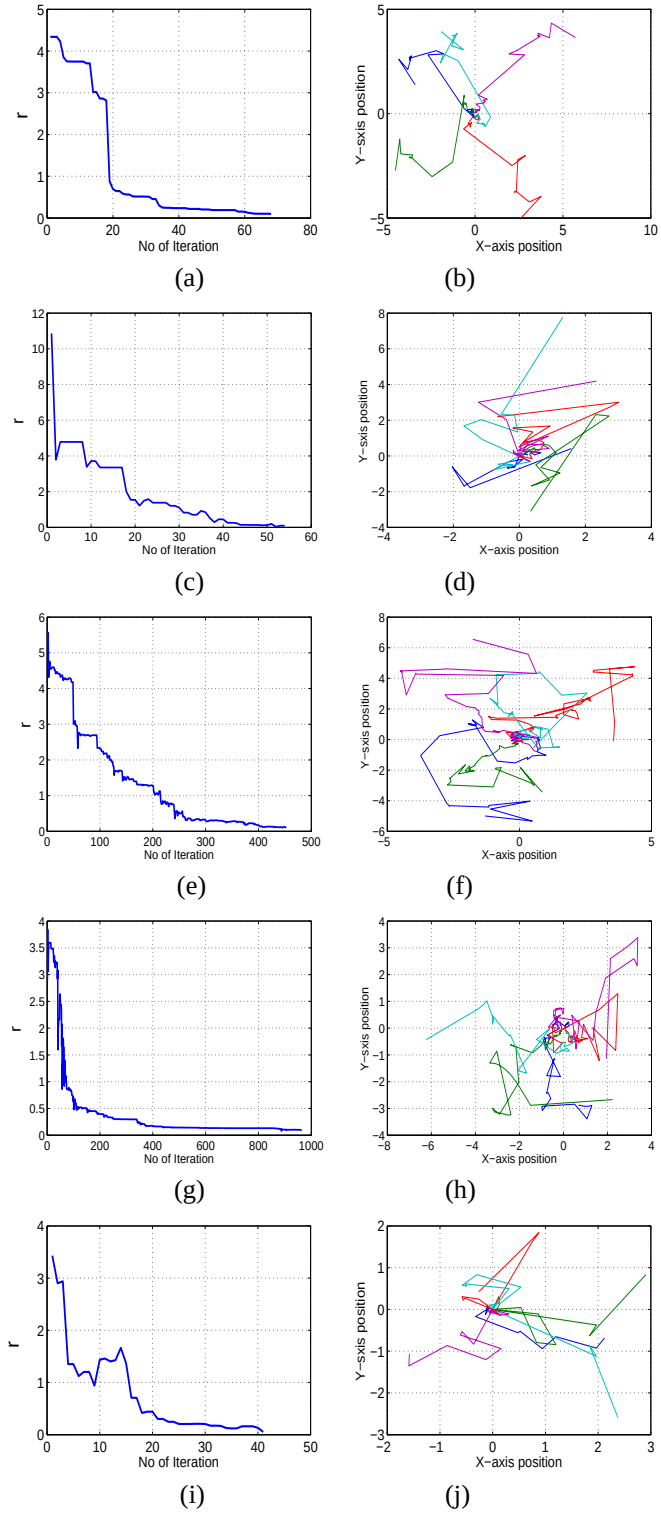


Figure 12: Final point is specified : (a) Algorithm A : r vs iteration (b) Algorithm A : trajectory of the agents (c) Algorithm (A, F_O) : r vs iteration (d) Algorithm (A, F_O) : trajectory of the agents (e) Algorithm (A, F_N) : r vs iteration (f) Algorithm (A, F_N) : trajectory of the agents (g) Algorithm F_O : r vs iteration (h) Algorithm F_O : trajectory of the agents (i) Algorithm F_N : r vs iteration (j) Algorithm F_N : trajectory of the agents

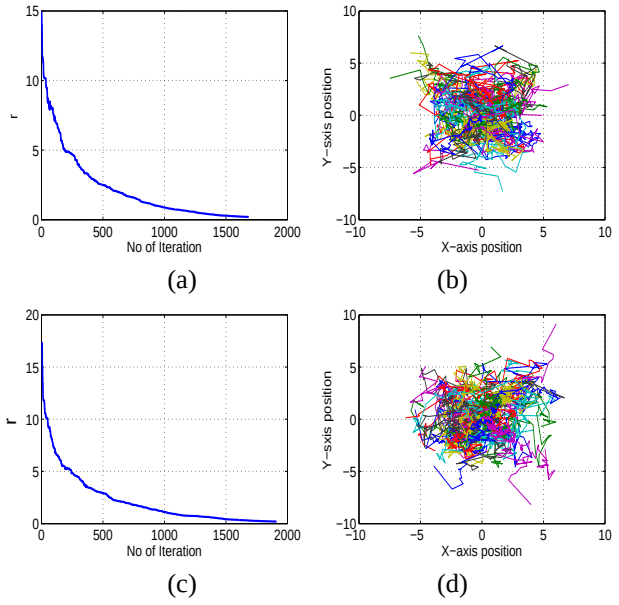


Figure 13: Final point is specified : 50 agents (a) Algorithm (A, F_O) : r vs iteration (b) Algorithm (A, F_O) : trajectory of the agents (c) Algorithm F_O : r vs iteration (d) Algorithm F_O : trajectory of the agents

Table 8: Comparison between algorithm for 20 agents with specified point (NC : No Convergence)

	20 Agents	
Algorithm	Iterations	Computation Time
A	NC	NC
(A, F_O)	554	24
(A, F_N)	NC	NC
F_O	494	23
F_N	NC	NC

8. CONCLUSIONS

In this paper we present some modified broadcast control algorithms. The results shows that some algorithms are effective for higher number of agents. We have also shown that without perturbation in the orientation, positional consensus is possible by setting randomized flags.

REFERENCES

[1] T. Vicsek, A. Czirok, E. B. Jacob, I. Cohen and O. Schochet. Novel type of phase transistions in a system of self-driven particles. *Physics Review Letter*, 75:1226–1229, 1995.

[2] G. Antonelli and S. Chiaverini. Kinematic control of platoons of autonomous vehicles. *IEEE Trans. Robotics and Automation*, 22(6) : 1285–1292, 2006.

[3] B. V. Gervasi and G. Prencipe. Coordination without communication: The case of the flocking problem. *Discrete Applied Mathematics*, 143: 203-223, 2004.

[4] M. Cieliebak, P. Flocchini, G. Prencipe and N. Santoro. Solving the robots gathering problem. *In Automata, Language and Programming*, Springer Verlag, Berlin, 2003.

[5] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots. *SIAM J. on Computing* , 28 : 1347-1363, 1999.

[6] T. Bretl. Control of Many Agents Using Few Instructions. *In Third Robotics Science and System Conference* , Atlanta, 2007.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[8] B.R. Donald, C.G. Levey, C.D. Mcgray, I. Paprotny and D. Rus. An untethered,electrostatic, globally controllable MEMS micro-robot. *J. Microelectromech. Syst*, 15(1) : 1-15 , 2006.

[9] K. Das and D. Ghose. Positional Consensus in Multi-Agent Systems using a Broadcast Control Mechanism. *In American Control Conference*, St. Losis, 2009, (to appear).

[10] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, U.S.A, 1997.

[11] D.G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Springer, Berlin, 2008.

[12] H. Ando, Y. Oasa and I. Suzuki. Distributed Memoryless Point Convergence Algorithm for Mobule Robots with Limited Visibility. *IEEE Tran. of Robotics and Automation*, 15(5): 818-828 , 1999.

[13] W. Rudin. *Principles of Mathematicla Analysis*. McGRAW-HILL Book Company, Sigapore, 1976.

[14] A. Jadabaie, J. lin, and A.S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbour rules. *IEEE Trans. Automat. Contr*, 48(6) : 988-1001 ,2003.

[15] G.C. Chasparis and J.S. Shamma. Linear-Programming-Based Multi-Vehicle Path Planning with Adversaries. *In American Control Conference*, Portland, 2005.

[16] C. Reynolds. Flocks, herd and schools: A distributed behavioral model. *Comput. Graph.*, 21(4) : 25-34 , 1984.

[17] V.D. Blondel, J.M. Hendrickx, A. Olshevsky and J.N. Tsitsiklis. Convergence in Multiagent Coordination, Consensus, and Flocking. *In Conference on Decision and Control*, Seville, 2005.

[18] J.K. Sengupta, G. Tintner and C. Millham. On some Theorems of Stochastic Linear Programming with Applications. *Management Science*, 10(1) : 143-159 , 1963.

Kaushik Das obtained a B.E. degree in Electrical Engineering from Bengal Engineering College (a D.U.), Calcutta, India in 2003 and ME degree in Electrical engineering from Jadavpur University, Calcutta, India in 2006. Presently, he is a research scholar in the Department of Aerospace Engineering at the Dynamics and Control System Lab.

Debasish Ghose obtained a B.Sc. (Engg.) in Electrical Engineering from the Regional Engineering College, Rourkela, India, in 1982, and the ME and PhD. degrees, also in electrical engineering, from the Indian Institute of Science, in 1984 and 1990. He is a professor in the Department of Aerospace Engineering at the Indian Institute of Science. His research interest are in the areas of guidance of aerospace vehicle, autonomous guided vehicles, applied game theory, multi-agent dynamics and control, decision-making in large scale systems and resource allocation and scheduling Problems.

Multi-Agent Rendezvous Algorithm with Rectilinear Decision Domain

Kaushik Das* and Debasish Ghose

GCDSL, Dept. of AE,
Indian Institute of Science,
Bangalore, India
{kaushikdas,dghose}@aero.iisc.ernet.in

Abstract. The aim of this paper is to develop a computationally efficient decentralized rendezvous algorithm for a group of autonomous agents. The algorithm generalizes the notion of sensor domain and decision domain of agents to enable implementation of simple computational algorithms. Specifically, the algorithm proposed in this paper uses a rectilinear decision domain (RDD) as against the circular decision domain assumed in earlier work. Because of this, the computational complexity of the algorithm reduces considerably and, when compared to the standard Ando's algorithm available in the literature, the RDD algorithm shows very significant improvement in convergence time performance. Analytical results to prove convergence and supporting simulation results are presented in the paper.

Keywords: Multi-agent, Rendezvous, Decision domain, Consensus.

1 Introduction

Research on multiple agents in the context of robotics is motivated by the fact that instead of using a highly sophisticated and expensive automated agent, it may be advantageous to use a group of small, simple, and relatively cheap autonomous agents (mobile robots or UAVs).

The autonomous control of multi agents has emerged as a challenging problem. The agents are assumed to have limited sensor and communication range and execute some local rule-based strategy depending on the information collected by each agent from the environment and from neighboring agents. One of the generic tasks that such a system of agent is often called upon to perform is to physically bring all the agents to a common point. This is called a multi-agent rendezvous problem [1]. This problem is important because if rendezvous is feasible, then more general formations are also achievable [2]. Previous work in this area are by Ando et al. [3] and Lin et al. [4] where each agent moves toward the rendezvous point by performing a sequence of "stop-and-go" moves. The *stop* mode is basically the *sensing period* and is an interval of fixed length. In the *go* mode the agents will maneuver in an interval of variable length and will move from its current position to a new position. In [3], as also in our paper, the sensing period

* The authors gratefully acknowledge the AOARD/AFOSR for their grant to this project.

P. Vadakkepat et al. (Eds.): FIRA 2010, CCIS 103, pp. 162–169, 2010.

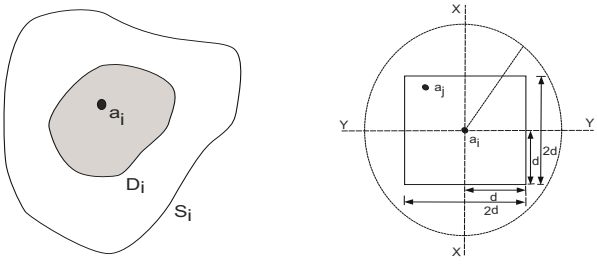
is assumed to be zero. Both [3] and [4] use algorithms that require determination of the smallest circle that contains a given set of agent positions. These algorithms are called "circumcenter algorithms". Although the complexity of this algorithm is proved to be subexponential of order $O(ne^{(2+o(1))\sqrt{2\ln n}})$ [5], the number of actual computations is fairly high. In our paper we generalize the notion of sensing domain and decision domain and show that by using a rectilinear decision domain the computations can be simplified considerably, thus bringing down the convergence time.

We show that our algorithm is far superior in terms of computational time than Ando's algorithm [3] which is the standard algorithm in the literature. we consider point robots with simple kinematics and instantaneous directional motion as in [3], [4].

2 Preliminaries

Let $R = \{a_1, a_2, \ldots, a_n\}$ be the set of robots or agents. The positions of agent a_i is given by $p_i = (x_i, y_i) \in \mathbb{R}^2$. The sensor domain of a agent a_i is denoted as S_i and its decision domain is denoted by D_i, where $D_i \subset S_i$. Information sensed from the decision domain is used to implement the algorithm. Essentially, we introduce the concept that information from the whole of the sensor domain need not be used for decision-making. In Fig. 1a, we give a schematic of these concepts. Note that, in general, p_i need not be inside S_i. An agent determines its set of neighbouring agents based on D_i. In this paper we assume that the sensing domain (S_i) of all the agents is circular with radius r. The decision domain (D_i) is a square of side $2d$, with $d < \frac{r}{\sqrt{2}}$, aligned with a pre-specified global (X,Y) reference frame. This is shown in Fig. 1b. The set of neighbors of agent a_i is defined as $N_i = \{a_j \mid (|x_i - x_j|) \leq d \text{ and } (|y_i - y_j|) \leq d\}$.

Note that an agent is also its own neighbor, so $a_i \in N_i$. Also, if $a_j \in N_i$ then $a_i \in N_j$. In Ando et al. [3] S_i and D_i are the same and are circles.



(a) (b)

Fig. 1. (a) General sensor domain and decision domain (b) The sensor domain is circular and the decision domain is a square aligned with the global $X-Y$ coordinates

3 Rectilinear Decision Domain (RDD) Algorithm

Convergence to a rendezvous point in [3] is proved through two properties: (i) Agents who are neighbours remain as neighbours (ii) Agents come closer with each other in

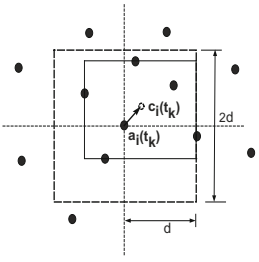


Fig. 2. Agent a_i will move to the centroid of the rectangle. The square with broken lines is the decision domain of agent a_i and agents inside it belong to N_i. The solid rectangle is the smallest rectangle that contains the agents in N_i.

some sense till they meet at a point. The RDD (rectilinear decision domain) algorithm retains these two ideas. Again as in [3] the RDD algorithm has the assumption that the initial graph is connected.

Algorithm RDD *(Rectilinear Decision Domain)*

Step 1: Each agent a_i determines its neighbour set N_i using its decision domain D_i.
Step 2: Each agent constructs the smallest rectangle, aligned with the global coordinate axes, that contains all the members of its neighbour set.
Step 3: Each agent computes the centroid of the rectangle and moves to it.

Fig. 2 illustrates these steps where $c_i(t_k)$ is the centroid of the rectangle at the time instant t_k. These steps are similar to Ando's algorithm [3], but for a few significant differences. In Step 1, Ando's algorithm determines neighbours using the sensor domain S_i. In Steps 2 and 3, Ando's algorithm computes the circumcenter of the neighbours and moves *toward* it subject to a constraint. Unlike RDD which allows the agents to move directly to the centroid, Anod's algorithm may not allow the agents to reach the centroid. These two important differences lead to high computational complexity, and thus higher convergence time, in Ando's algorithm. In RDD, an agent a_i uses the information $P_i = \{(x_j, y_j) | a_j \in N_i\}$ where $a_j \in N_i$, and computes $\max\{x_j\}$, $\min\{x_j\}$, $\max\{y_j\}$ and $\min\{y_j\}$ to obtain the rectangle. The computational complexity of this operation is $O(n)$. We will now state two important theorems.

Theorem 1. *An agent's movement will be confined to a square of side d centered at the agent's current position and aligned with the global reference frame.*

Proof. Consider the maximum deviation of the centroid of the rectangular area along the X-axis from the agent's current position. This will be less than $d/2$ because the maximum deviation of a neighbor's position along the X-axis is d. Similar arguments hold for the Y-axis. So the agent movement will be confined within a square of side d, centered at the agent's current position. □

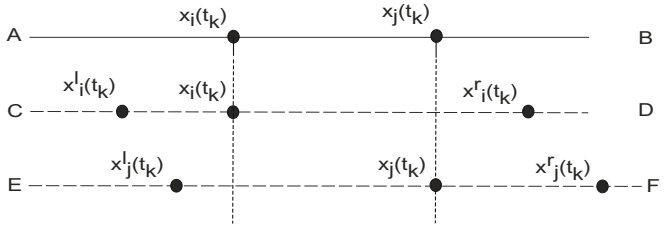


Fig. 3. Agent a_i,a_j and its neighbor projection along X-axis

Theorem 2. *If at any time t_k, agents a_i and a_j are neighbors, then they will be neighbors for all time $t > t_k$.*

Proof. Let the projections of agents a_i and a_j on the X axis at time t_k be denoted by $X_i(t_k)$ and $X_j(t_k)$, respectively, where $X_j(t_k) > X_i(t_k)$. Let the left most and right most projections of the neighbors of agent a_i be $X_i^l(t_k)$ and $X_i^r(t_k)$, respectively. The position of agents a_i and a_j on the X axis at time instant t_{k+1} is then $X_p(t_{k+1}) = \frac{X_p^l(t_k)+X_p^r(t_k)}{2}$; where $p = i, j$. The distance between the agents a_i and a_j along the X axis at time instant t_{k+1} is $\mid X_j(t_{k+1}) - X_i(t_{k+1}) \mid = \frac{1}{2} \mid (X_j^l(t_k) - X_i^l(t_k)) + (X_j^r(t_k) - X_i^r(t_k)) \mid$.

We can show that $\mid X_j^l(t_k) - X_i^l(t_k) \mid \leq d$, since when $X_i^l(t_k) \leq X_j^l(t_k) \leq X_i(t_k)$, we can write $0 \leq X_j^l(t_k) - X_i^l(t_k) \leq X_i(t_k) - X_i^l(t_k) \leq d$. Again, when $X_j^l(t_k) \leq X_i^l(t_k) \leq X_i(t_k)$, we can write $X_j^l(t_k) - X_i^l(t_k) \leq 0 \leq X_i(t_k) - X_i^l(t_k) \leq d$. Similarly, we can prove $\mid X_i^r(t_k) - X_j^r(t_k) \mid \leq d$. This implies that $X_j(t_{k+1}) - X_i(t_{k+1}) \leq d$. We can prove the same along the Y axis too. □

The theorem proves that connectivity between agents, which plays an important role in the performance of the algorithm, is preserved.

4 Analysis of the RDD Algorithm

Let the global convex hull made by the positions of the agents at the time instant t_k be denoted by $Co(t_k)$. We can define the diameter of the convex hull at the time instant t_k as $dia(Co(t_k)) = max\{\parallel p_i(t_k) - p_j(t_k) \parallel\}, \;\; i, j \in \{1, 2, \ldots, n\}$. When rendezvous is achieved the diameter of the global convex hull is zero. We will first show that the diameter of the global convex hull will reduce at each step.

Let us consider $Co_i(t_k)$ as the convex hull made by the neighbor set of the agent a_i at the time instant t_k. Let the smallest rectangle containing the neighbor set of agent a_i, and aligned along the global coordinate axes, at time instant t_k be $R_i(t_k)$. It is obvious that $Co_i(t_k) \subset R_i(t_k)$. Let $y^i_{min} = \min\{y_j\}$, where y_j is the $y-$th coordinate of $a_j \in N_i$. Other variables are similarly defined. Then we have the following result.

$$mid(R_i(t_k)) = (1/2)((x^i_{min} + x^i_{max}), (y^i_{min} + y^i_{max})) \tag{1}$$

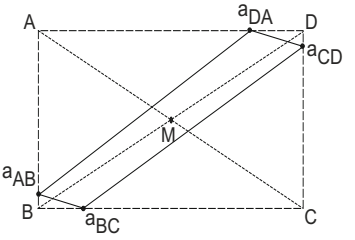


Fig. 4. Rectangle made by the neighbor of agent a_i

Theorem 3. *mid* $(R_i(t_k)) \in Co_i(t_k)$ *and mid* $(R_i(t_k))$ *is not an corner point of* $Co_i(t_k)$

Proof. Consider Fig. 4. Here $R_i(t_k)$ is the rectangle *ABCD* and mid point of it is *M*. Each side must have an agent. Let N_{AB} be the set of agents on side *AB*. Similarly, N_{BC}, N_{CD} and N_{DA} are the set of agents on the side *BC*, *CD* and *DA*, respectively. It is clear that $N_{AB} \cap N_{CD} = \emptyset$ and $N_{BC} \cap N_{DA} = \emptyset$. Choose agents a_{AB}, a_{BC}, a_{CD} and a_{AB} in such a way that $a_{AB} \in N_{AB}$, $a_{BC} \in N_{BC}$, $a_{CD} \in N_{CD}$ and $a_{DA} \in N_{DA}$. The line joining a_{AB} and a_{BC} will separate *B* and *M*. Similarly, the line joining a_{BC} and a_{CD}will separate *B* and *M*, the line joining a_{CD} and a_{DA} will separate *D* and *M* and line joining a_{DA} and a_{AB} will separate *A* and *M*. So, we can write $M \in Co\langle a_{AB}, a_{BC}, a_{CD}, a_{DA}\rangle$. As $Co\langle a_{AB}, a_{BC}, a_{CD}, a_{DA}\rangle \subset Co_i(t_k)$, then we can write $M \in Co_i(t_k)$. Let, the coordinate of the mid $(R_i(t_k))$ is (x^i_{mid}, y^i_{mid}). If the rectangle is not a point then it is evident $x^i_{mid} \neq x^i_{min}$, $x^i_{mid} \neq x^i_{max}$, $y^i_{mid} \neq y^i_{min}$ and $y^i_{mid} \neq y^i_{max}$. This observation and $M \in Co_i(t_k)$implies that *M* can not be a corner point of $Co_i(t_k)$. □

Now consider any agent a_i. Let the maximum distance along the *X*-axis on the right side between agent a_i and its neighbors be d^i_{rx} and on the left side is d^i_{lx}. Similarly, along the *Y*-axis the maximum distance above a_i is d^i_{ay} and below of a_i is d^i_{by}.The position of the agent a_i at time instant (t_{k+1}) will be

$$x_i(t_{k+1}) = x_i(t_k) - (d^i_{lx} - d^i_{rx})/2;\;\; y_i(t_{k+1}) = y_i(t_k) - (d^i_{by} - d^i_{ay})/2 \tag{2}$$

The movement of the agent a_i will depends upon $\{d^i_{lx}, d^i_{rx}, d^i_{ay}, d^i_{by}\}$. The agent a_i will be stationary if $d^i_{lx} = d^i_{rx}$ and $d^i_{ay} = d^i_{by}$. Next we will state a theorem that agents at the corner points global convex hull $Co(t_k)$ cannot remain stationary.

Theorem 4. *For any agent* a_i *which is at the corner of* $Co(t_k)$ *and has at least one non-located neighbor, both* $d^i_{lx} = d^i_{rx}$ *and* $d^i_{ay} = d^i_{by}$ *cannot be satisfied.*

Proof. We will prove this by contradiction. Let $d^i_{lx} = d^i_{rx}$ and $d^i_{ay} = d^i_{by}$ for an agent a_i which is at a corner point. Let the agent a_i be at corner point *A* in Fig. 5. The neighbors a_j and a_k for which $d^i_{lx} = d^i_{rx}$ and $d^i_{ay} = d^i_{by}$ is at *B* and *C*, respectively. From Fig. 5, we can write $d^i_{rx} = d_1 \sin\theta_1$, $d^i_{lx} = d_2 \sin\theta_2$, $d^i_{by} = d_1 \cos\theta_1$ and $d^i_{ay} = d_2 \cos\theta_2$. If $d^i_{rx} = d^i_{lx}$ and $d^i_{ay} = d^i_{ay}$,

$$d_1/d_2 = \sin\theta_2/\sin\theta_1 = \cos\theta_2/\cos\theta_1 \tag{3}$$

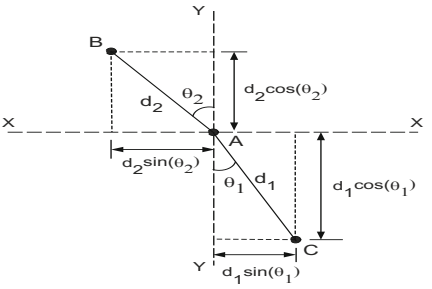


Fig. 5. Agent a_i is at the corner point of $Co(t_k)$

Equating the above two equations $\theta_2 = \theta_1$. Thus, A will be on the line BC. So, A is not a corner point. This leads to the contradiction. □

From Theorem 4 and Theorem 3, it is clear that agents at the corner of the convex will move either inside of the convex hull or move along the edges of the convex hull.

Theorem 5. *No agent can be at a corner point of the convex hull $Co(t_k)$ at time t_{k+1}.*

Proof. We prove this by contradiction. Suppose an agent a_i reaches the corner point c_1 of the convex hull $Co(t_k)$ at time t_{k+1}. This implies that c_1 is the mid point of the rectangle made by the neighbors of agent a_i at time t_k. Then, according to Theorem 3, c_1 is inside the convex hull $Co_i(t_k)$. Again $Co_i(t_k) \subseteq Co(t_k)$. This implies that c_1 is the convex combination of some points of $Co(t_k)$. This leads to the contradiction. □

Theorem 6. $Co(t_{k+1}) \subset Co(t_k)$

Proof. Let there be m corner points of the convex hull $Co(t_k)$ given by $P_c(t_k) = \{p_1, \ldots, p_m\}$ and m_1 corner points of the convex hull $Co(t_{k+1})$ given by $P_c(t_{k+1}) = \{\widehat{p}_1, \ldots, \widehat{p}_{m_1}\}$. Now, $P_c(t_k) \cap P_c(t_{k+1}) = \emptyset$, because no agent can reach at the corner of the convex hull $Co(t_k)$ according to lemma 5 at time instant t_{k+1}. The corner point agents of $Co(t_k)$ must move (according to Theorem 4). Since $\widehat{p}_i \in Co(t_k)$ and $p_i \notin Co(t_{k+1})$, for all $p_i \in P_c(t_k)$, we have $Co(t_{k+1}) \subset Co(t_k)$. □

Next we will show that the diameter of the global convex hull will reduce at each step.

Theorem 7. $dia(Co(t_{k+1}) < dia(Co(t_k))$

Proof. As before, let $P_c(t_k) = \{p_1, \ldots, p_m\}$ be the set of corner points of $Co(t_k)$ and $P_c(t_{k+1}) = \{\widehat{p}_1, \ldots, \widehat{p}_{m_1}\}$ be the set of corner points of $Co(t_{k+1})$. According to the definition of the diameter of a convex hull[7]

$$dia(Co(t_k)) = \max\{\|p_i - p_j\|, \forall p_i, p_j \in P_c(t_k)\} \tag{4}$$

$$dia(Co(t_{k+1})) = \max\{\|\widehat{p}_i - \widehat{p}_j\|, \forall \widehat{p}_i, \widehat{p}_j \in P_c(t_{k+1})\} \tag{5}$$

Again $\|p_i - p_j\| < dia(Co(t_k))$ for all $p_i, p_j \in Co(t_k)$ and $p_i, p_j \notin P_c(t_k)$. Here $P_c(t_{k+1}) \in Co(t_k)$ and $P_c(t_{k+1}) \cap P_c(t_k) = \emptyset$. From this, we get $dia(Co(t_{k+1})) < dia(Co(t_k))$. □

The above theorem tells us that the sequence of convex hulls, generated by the positions of the agents will make a descending chain of convex sets. Under similar arguments as Ando et al.[3] it can be shown that the convex hull will become a point as all the agents converge to a point. We omit the proof due to space limitations.

5 Simulation Results and Implementation Issues

In Fig. 6, four snapshots for 10 agents have been shown. One can see that all the agents eventually converge to a point. The system converges when the maximum distance between the agents along the X axis and Y axis is less than the decision domain distance (d), since the agents would converge to a single point in the very next step.

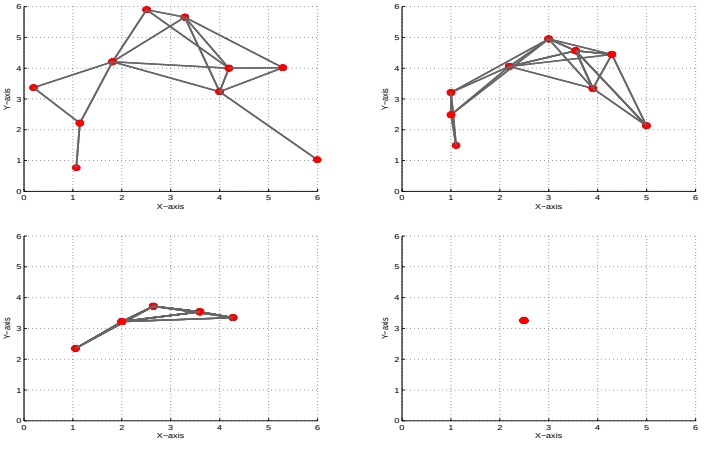


Fig. 6. 10 agents placed randomly converge to a point

The comparative study between Ando's algorithm and the RDD algorithm is executed. In Table 1, the comparison of computational time to converge is given. The results shows that RDD algorithm is superior in terms of computational time.

In Table 2 comparison of the number of iterations to converge is given. The iteration number in case of RDD algorithm is a bit higher than Ando's algorithm. The reason behind for this is the decision domain in case of RDD algorithm is smaller than Ando's algorithm.

The algorithm can be implemented by any agent (robot) that can gather information about and from its neighbors (positions and orientations) using sonar sensors, vision sensors or laser scanners. After generating the control command the agents can broadcast the control command with positions and orientation of the centroid information by a short range communication mechanism. An agent receiving a broadcast control command from another agent which is not in its decision domain will ignore the command.

Table 1. Comparison of Computational Time

Time to Converge (sec)			
Number of Agents	Ando	RDD	Ando/RDD
20	0.5856s	0.0329	17.8065
50	3.0677s	0.1758	17.4544
100	12.1486	0.5439	22.3360
150	40.7101	1.7395	23.4031

Table 2. Comparison of Number of Iterations

Number of Iterations to Converge				
Number of Agents	Ando	RDD	Ando (Time/Iteration)	RDD (Time/Iteration)
20	3.8000	5.2667	0.1541	0.0062
50	6.8667	9.3333	0.4468	0.0188
100	8.4667	11.8667	1.4349	0.0458
150	11.3333	15.9333	3.5921	0.1092

6 Conclusions

We presented and analyzed a rendezvous algorithm considering a rectilinear decision domain. The computational complexity of RDD algorithm is low compared to the well established Ando's algorithm in the literature. The RDD algorithm is simpler in terms of few computation needed and in relaxing the restriction on the movement of the agents to the centroid of the rectangle. In this sense, it is a purely centroidal algorithm. The simulation results support our claim.

References

1. Kranakis, E., Krizanc, D., Rajsbaum, S.: Mobile Agent Rendezvous: A Survey. In: Flocchini, P., Gąsieniec, L. (eds.) SIROCCO 2006. LNCS, vol. 4056, pp. 1–9. Springer, Heidelberg (2006)
2. Lin, Z., Broucke, M., Francis, B.: Local Control Strategies for Groups of Mobile Autonomous Agents. IEEE Transactions on Automatic Control 49, 622–628 (2004)
3. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed Memoryless Point Convergence for Mobile Robots with Limited Visibility. IEEE Transactions on Robotics and Automation 15, 818–828 (1999)
4. Lin, J., Morse, A.S., Anderson, B.D.O.: The Multi-Agent Rendezvous problem. In: Conference on Decesion & Control, Maui, Hawaii, USA, pp. 1508–1513 (2003)
5. Gartner, G.: A subexponential algorithm for abstarct optimization problems. SIAM Journal Computers 24(5), 1018–1035 (1995)
6. Das, K., Ghose, D.: Positional Consensus in Multi-Agent Systems using a Broadcast Control Mechanism. In: American Control Conference, St. louis, Missouri, USA, pp. 5731–5736 (2009)
7. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1972)

Positional Consensus of Multi-Agent Systems Using Linear Programming Based Decentralized Control with Rectilinear Decision Domain

Kaushik Das* and Debasish Ghose

GCDSL, Dept. of AE,
Indian Institute of Science,
Bangalore, India
{kaushikdas,dghose}@aero.iisc.ernet.in

Abstract. In this paper we develop a Linear Programming (LP) based decentralized algorithm for a group of multiple autonomous agents to achieve positional consensus. Each agent is capable of exchanging information about its position and orientation with other agents within their sensing region. The method is computationally feasible and easy to implement. Analytical results are presented. The effectiveness of the approach is illustrated with simulation results.

Keywords: Multi-agent, Linear programming, Consensus, Decentralize.

1 Introduction

In many applications teams of robots or autonomous agents are being used to achieve a mission that was earlier entrusted to a single autonomous robot. Such systems have the obvious advantage of redundancy, architectural simplicity, and low cost. But they are more difficult to control as they have to be controlled autonomously through a decentralized control strategy. The main goal of control strategies for groups of autonomous agents is to achieve a coordinated objective while using local information. One of the problems that is of great importance in this field is that of achieving consensus, that is, achieving identical values for some specified subset of the states of the agents. For instance, the agents may try to converge to the same direction of movement [1] after some time or they might want to converge to a point. Both are problems in achieving consensus. If we have a centralized system with perfect information then achieving consensus is a trivial matter, since the central controller can instruct each agent suitably to reach a common consensus point. The challenging task is to achieve consensus using a distributed and decentralized strategy. Several classes of problems in the area has been addressed in the literature in [2] - [5].

In [6] a *Linear Programming (LP)* based centralized formulation was proposed to achieve positional consensus of multiple agents. The central controller observes the position and orientation of the agents and, using this information, it computes a common control for all the agents based on a *Linear Programming (LP)* technique. The central

* The authors gratefully acknowledge the AOARD/AFOSR for their grant to this project.

P. Vadakkepat et al. (Eds.): FIRA 2010, CCIS 103, pp. 178–185, 2010.

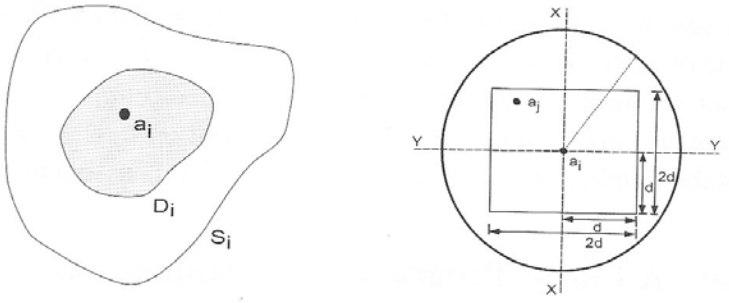


Fig. 1. (a) General sensor domain and decision domain (b) The sensor domain is circular and the decision domain is a square aligned with the global $X-Y$ coordinates

controller broadcasts identical control command to all the agents. The agents implement the common control command in their own local reference frame to move toward consensus. It was shown that perfect consensus is not possible by this method when the number of agents is more than two. So, a post-command randomization was introduced for the agents. Using this technique it was shown that consensus for larger number of agents could be achieved. However, the computation of the command was done in a centralized framework.

In this paper we show that it is possible to achieve positional consensus in a *decentralized* framework, using a similar LP based formulation. We assume the agents' decision domain in $\mathbb{R}^2$ space agent is a square. We assume that agents are able to communicate with each other within the sensing zone and they are also able to exchange information about their position and orientation. Using these local information, each agent will run the LP based algorithm and will compute a control for itself. After computing the control command, an agent can directly implement it or it can obtain information from its neighbors about their control commands and then implement an average control. We show that the decentralized strategy leads to consensus.

2 Preliminary Definitions

Let $R = \{a_1, a_2, \ldots, a_n\}$ be the set of robots or agents. The positions of agent a_i is $x_i = (p_{i1}, p_{i2}) \in \mathbb{R}^2$ and the orientations is $\theta_i \in (-\pi, \pi]$. The sensor domain of an agent a_i is denoted as S_i and its decision domain is denoted by D_i, where $D_i \subset S_i$. Information sensed from the decision domain is used to implement the algorithm. Essentially, we introduce the concept that information from the whole of the sensor domain need not be used for decision-making. In fact, we can select decision domains that make computations simpler. In Fig. 1a, we give a schematic of these concepts. Note that, in general, p_i need not be inside S_i. An agent determines its set of neighbouring agents based on D_i. In this paper we assume that the sensing domain (S_i) of all the agents is circular with radius r. The decision domain (D_i) is a square of side $2d$, with $d < \frac{r}{\sqrt{2}}$, aligned with a pre-specified global (X,Y) reference frame. This is shown in Fig. 1b. The set of neighbors of agent a_i is defined as $N_i = \{a_j \mid (|x_i - x_j|) \leq d;\ (|y_i - y_j|) \leq d;\ \ i \neq j\}$.

This system can be represented by a proximity graph $G(t) = (V, E(t))$, where V denotes the node or vertex of the graph. Each node represents an agent or a robot. $E(t)$ represents the set of edges between the nodes at any time. Let $a_i \sim a_j$ denote the edges between the agents a_i and a_j and $(a_i\, a_j) \in E(t)$ if and only if $a_j \in N_i$. The degree of any node represents the number of neighbors of the agents that is associated with that node.

3 Algorithm: A Linear Programming Formulation

The LP based centralized consensus algorithm is described in [6]. In which a central controller computes the control command using the position and orientation of the agents and the control is broadcast to all the agents. The computed control will be in $(d, \Delta\theta)$ form. After receiving this control command, each agent will execute a rotation of angle $\Delta\theta$ and translation d. Here, we present an algorithm which is *decentralized* in nature.

Let the set of positions and orientation of the neighbors of agent a_i be $\{\{x_j\}, \{\theta_j\}\}$, where $a_j \in N_i$. As in [6], we define the control command computed by agent a_i as $(\Delta\theta_i, d_i)$ using information from its neighbors only. We define our performance measure as the half length, denoted by $r_i > 0$, of the side of a square oriented along the global coordinate frame, and containing all the final positions of the agent and its neighbors. Let this square be centered at $z_i = (z_{1_i}, z_{2_i}) \in \mathbb{R}^2$. Since all neighbors of agent a_i execute the same command $(\Delta\theta_i, d_i)$, their final positions, given by $x_{jf} = [q_{i1}\ \ q_{i2}] \in \mathbb{R}^2$, are

$$\begin{bmatrix} q_{i1} \\ q_{i2} \end{bmatrix} = \begin{bmatrix} p_{i1} \\ p_{i2} \end{bmatrix} + \begin{bmatrix} \cos\theta_{i0} & -\sin\theta_{i0} \\ \sin\theta_{i0} & \cos\theta_{i0} \end{bmatrix} \begin{bmatrix} u_{1_i} \\ u_{2_i} \end{bmatrix} \tag{1}$$

where, $u_{1_i} = d_i \cos(\Delta\theta_i)$ and $u_{2_i} = d_i \sin(\Delta\theta_i)$ are the control variables used. Note that Eqn. (1) is a set of linear equations. Now, we formulate the linear programming problem for agent a_i as,

Minimize r_i
Subject to

$$-r_i \le p_{j1} + u_{1_i}\cos\theta_j - u_{2_i}\sin\theta_j - z_{1_i} \le r_i \tag{2}$$

$$-r_i \le p_{j2} + u_{1_i}\sin\theta_j + u_{2_i}\cos\theta_j - z_{2_i} \le r_i \tag{3}$$

$$r_i \ge 0; \quad j \in N_i \tag{4}$$

This is a linear programming problem with the decision vector as $(r_i, z_{1_i}, z_{2_i}, u_{1_i}, u_{2_i})$. Note that the decision vector remains the same irrespective of the number of neighbors. Only the number of inequality constraint increases with the number of agents. Also, note that z_{1_i},z_{2_i},u_{1_i} and u_{2_i} are free variables and can take both +ve or -ve.

4 Decentralized Control

In this section we will give a formal description of the algorithm. It is obvious that the outcome of this algorithm depends upon the positions and orientations of the neighbors

of an agent as well as of its own. The positions of the agents a_i at time instant t_{k+1} can be written as

$$x_i(t_{k+1}) = x_i(t_k) + U(\Delta\theta_j(t_k), d_j(t_k)) \tag{5}$$

where, $U(\Delta\theta_j, d_j) = f(x_j t_k, \theta_j t_k)$.

While computing the control command, agent a_i may or may not include its own position and orientation. Also, after computing the control command it may or may not share its control with its neighbor. This gives four possible operation for implementing this algorithm. The four cases are shown in the Table 1. When agent will share control command with its neighbors, we will say that it uses a broadcast mechanisms (which is restricted to its neighbors) to communicate with its neighbor. When there is no sharing of control command then we will say that there is no broadcast communication. Note that broadcast based algorithm needs some effort in implementing the algorithm as the broadcast is restricted.

Table 1. Four different cases

	No Broadcast	Broadcast
Self Included $j \in N_i \cup \{i\}$	Case *A*	Case *B*
Self not Included $j \in N_i$	Case *C*	Case *D*

4.1 No Broadcast: Case *A* and Case *C*

The agents will share its positions and orientation but it will not share its control with its neighbor. It may or may not include its position and orientation while computing the control command. Under this paradigm there are two cases

1. Case A: Compute control using the information of the neighbor as well as of its own. Then, $U(\Delta\theta_j, d_j) = f(x_i(t_k), \theta_i(t_k))_{i \in N_j \cup \{j\}}$.
2. Case C: Compute control using the information of the neighbor only. $U(\Delta\theta_j, d_j) = f(x_i(t_k), \theta_i(t_k))_{i \in N_j}$

4.2 Broadcasting – Averaging Rule: Case *B* and Case *D*

The agents will send or broadcast its control to its neighbors. So a neighbor will have more than one control input. The agents will compute an average of all its control and will move to a new position. Here also two case can arise

1. Case B: Compute the control using information from neighbors as well as of its own and broadcast to the neighbors. Then, $U_{a_i} = \frac{1}{m_i+1}\left((\Delta\theta_i(t_k), d_i(t_k)) + \sum_{j \in N_i}(\Delta\theta_j, d_j)\right)$.

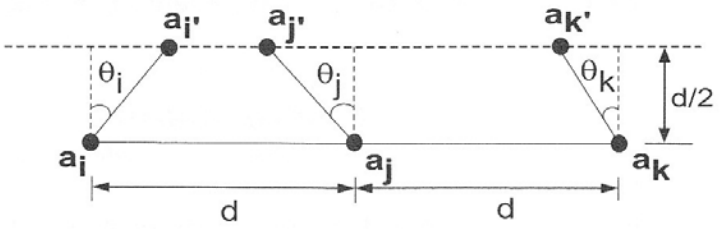


Fig. 2. An illusration of three agents

2. Case D: Compute control using only the neighbor information and broadcast to neighbors. Then, $U_{a_i} = \frac{1}{m_i}\left((\Delta\theta_i(t_k), d_i(t_k)) + \sum_{j\in N_i}(\Delta\theta_j, d_j)\right)$.

5 Modifications: Preserving Connectivity

Applying the algorithm has the problem that since the connectivity between agents may not be preserved, as will be seen from the following example.

Consider only three agents which are collinearly placed at intervals of distance d, i.e., the maximum sensing distance (see Fig. 2) along the $X-$axis. Without loss of generality we put a boundary at $d/2$ distance in the Y-axis. This will ensure that the connectivity along the Y-direction will not be violated. From Fig. 2 we can see that the initial graph is connected. The positions of the agents at the time instant t_{k+1} are a_i',a_j' and a_k'. The graph will not be connected if $\theta_k \geq \theta_j$ because $(d/2)\sin(\theta_k) \leq (d/2)\sin(\theta_j)$. We can see from the above result that the graph will not be connected using the above algorithms and positional consensus may not always be possible. To ensure positional consensus, we have to ensure the connectivity of the graph always.

In order to achieve connectivity, the movement of the agents should be restricted. We call this as pairwise motion constraint. This is shown in Fig.3(a). In this figure agent a_i has one neighbor a_j. The movement of agent a_i is restricted within the square. The center of the square is C_{ij} which is the mid point of the line joining agents a_i and a_j. The side length is d. This constraint ensures that if two agents connect at time t_k, then they remain connected for all time $t > t_k$.

In Fig.3(b) we can see that agent a_i has two neighbors a_j and a_k, g_i is the goal point of the agents, computed by the algorithm. Now the shaded region is the common overlapping zone between the two squares. For agent a_j, it can move towards the goal by a distance d_{ij} and for agent a_k it can move a distance d_{ik}. Eventually, the agent a_i will move the distance $d = \min\{d_{ij}, d_{ik}\}$ towards the goal. So, in general,an agent will move towards the goal $d = \min\{d_{ij}\}, a_j \in N_i$.

The common overlap zone is shown by the region $ABCD$ in Fig.4. The agent position at time instant t_k is $a_i(t_k)$. The goal positions is at $g_i(t_k)$ which is outside of the common overlap zone. The agent can move up to the point a_f which is at the border of the

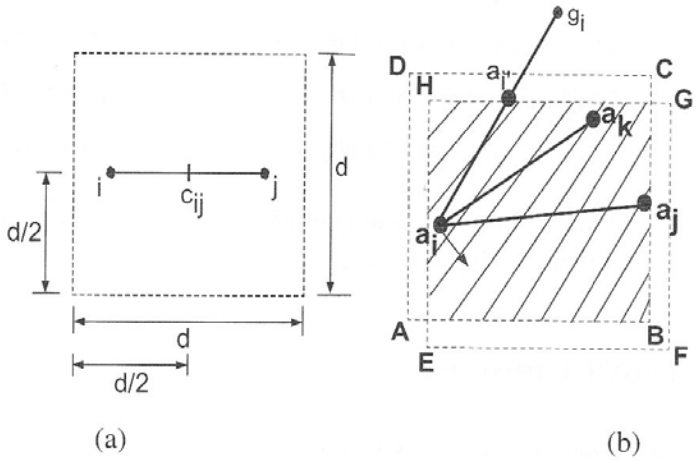


Fig. 3. An illustration of agents movement

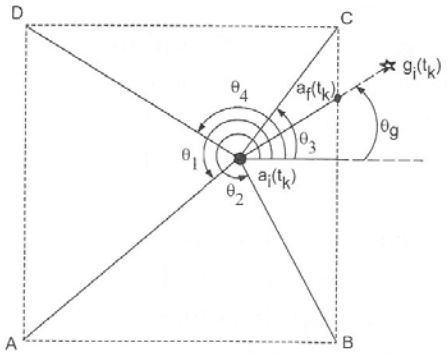


Fig. 4. Agent movement in common overlap Region

common overlap region. To compute the final positions of the agent a_i, we have divided the whole overlap zone into four parts.

1. $In_1 : \theta_g \in [0, \theta_1) \cup [\theta_4, 2\pi)$
2. $In_2 : \theta_g \in [\theta_1, \theta_2)$
3. $In_3 : \theta_g \in [\theta_2, \theta_3)$
4. $In_4 : \theta_g \in [\theta_3, \theta_4)$

Now we define four index function I_1, I_2, I_3 and I_4. The index I_1 can take values from $\{0, 1\}$. We can write

$$I_i = \begin{cases} 1 \ , \theta_g \in In_i \\ 0 \ , \theta_g \notin In_i \end{cases} ; \quad i \in \{1, \ldots, 4\}$$

In the four different cases the movement of the agents will be different. The final positions coordinate of the agents are,

$$p_{i1_f} = p_{i1} + (p_{2_{c_3}} - p_{i2}) * I_1 * \cot\theta_3 + (p_{2_{c_3}} - p_{i2}) * I_2 * \cot\theta_g$$
$$+(p_{2_{c_1}} - p_{i2}) * I_3 * \cot\theta_1 + (p_{2_{c_1}} - p_{i2}) * I_4 * \cot\theta_g \quad (6)$$
$$p_{i2_f} = p_{i2} + (p_{1_{c_3}} - p_{i1}) * I_1 * \tan\theta_g + (p_{1_{c_3}} - p_{i1}) * I_2 * \tan\theta_3$$
$$+(p_{1_{c_1}} - p_{i1}) * I_3 * \tan\theta_g + (p_{1_{c_1}} - p_{i1}) * I_4 * \tan\theta_1 \quad (7)$$

The above equations are valid when the goal point is outside common overlap region or on the boundary the common overlap region.

6 Achieving Perfect Consensus

The solution of the linear programming (LP) based algorithm will yield control instructions that can be broadcast to all the neighbors using which the agents will move to a new position. It can be shown that no further improvement of the performance (that is, the agent will be stationary) can be achieved by repeated use of the algorithm. The graph will not change if the set of neighbors do not change. In other words, repeated application of the LP based decentralized algorithm with the new final positions, will not yield any solution. Suppose we represent the LP algorithm as an operation L on the initial conditions that yields the solution as, $L(x_{i0}, \theta_{i0} | i = 1, \ldots, n) = (u_1^*, u_2^*, r*, x_{if}, \theta_{if})$.

If the neighbor set remains, then, $L(x_{if}, \theta_{if} | i = 1, \ldots, n) = (0, 0, r*, x_{if}, \theta_{if})$. This was the case in the centralized strategy [6]. However, in the decentralized case, it is possible that the neighbors set of an agent can change and will yield non-zero central commands for the agents at each successive applications of the strategy. But, subsequently when agents come sufficiently close the graph becomes completely connected and the control command becomes zero. In order to continuously get non zero control commands, we introduce a perturbation strategy given below.

In the *perturbed case* where the agents randomly perturb final orientation angle after each LP solution is now implemented as follows [6].

$$\hat{\theta}_{i,k+1} = \theta_{i,k+1} + \nu_{i,k+1} \quad (8)$$

In (8) $\nu_{i,k+1}$ represents the perturbation angle.

7 Simulations Results and Implementation Issues

In the first set of simulations we have considered 5 agents which are placed randomly at the $\mathbb{R}^2$ plane. We have set their orientation randomly. The result is shown in Fig. 5. We can see that using this algorithm all the agents eventually come to consensus.

The algorithm can be implemented by any agent (robot) that can gather information about and from its neighbors (positions and orientations) using sonar sensors, vision sensors or laser scanners. After generating the control command the agents can broadcast the control command with positions and orientation of the centroid information by a short range communication mechanism. An agent receiving a broadcast control command from another agent which is not in its decision domain will ignore the command.

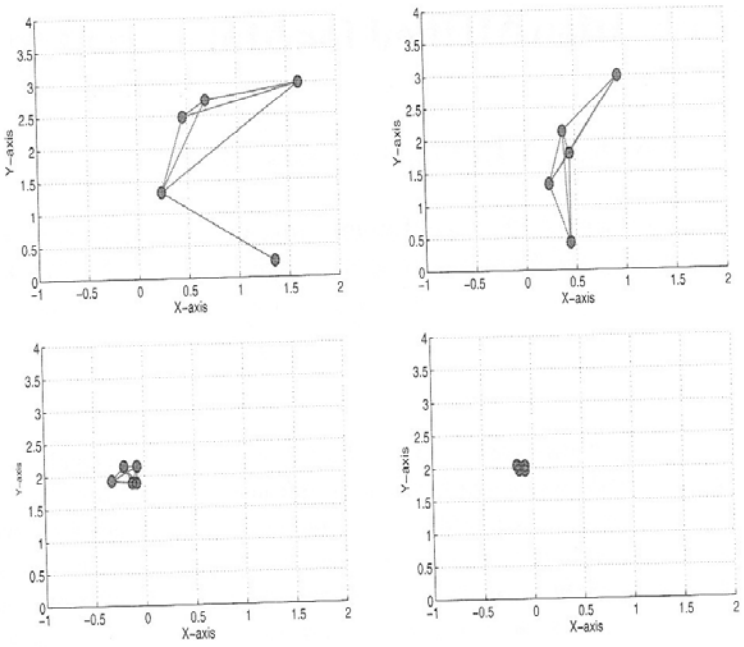


Fig. 5. Modified Algorithm: 5 agent preserves connectivity at each instant of time and come closer to a consensus

8 Conclusions

We presented a decentralize LP based algorithm. This work extends previous work on broadcast based centralized control of agents to a decentralized framework. The flexibility, that the algorithm gives to the individual agents allows us to design effective consensus algorithms. In future, we will focus on computational time and implementation issues.

References

1. Vicsek, T., Czirok, A., Jacob, E.B., Cohen, I., Schochet, O.: Novel type of phase transistions in a system of self-driven particles. Phys. Rev. Lett. 75, 1226–1229 (1995)
2. Jadababaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbour rules. IEEE Trans. Automat. Contr. 48(6), 988–1001 (2003)
3. Saber, R.O.: Flocking in multiagent dynamic systems: Algorithms and theory. IEEE Trans. Automat. Contr. 51 (2006)
4. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed Memoryless Point Convergence for Mobile Robots with Limited Visibility. IEEE Transactions on Robotics and Automation 15, 818–828 (1999)
5. Cortes, J., Martínez, S., Bullo, F.: Robust Rendezvous for Mobile Autonomous Agents via Proximity Graphs in Arbitrary Dimensions. IEEE Trans. Automat. Contr. 51, 1289–1298 (2006)
6. Das, K., Ghose, D.: Positional Consensus in Multi-Agent Systems using a Broadcast Control Mechanism. In: American Control Conference, St. louis, Missouri, USA, pp. 5731–5736 (2009)
7. Godsil, C., Royle, G.: Algebric Graph Theory. Springer, Heidelberg (2001)

ICIUS 2010
Nov 3-5, 20
Bali, Indone

Multi-Agent Rendezvous Algorithms under Various Information Paradigms

Kaushik Das* and **Debasish Ghose**

*Research Scholar, Guidance, Control and Decision System Laboratory Aerospace Engineering Department, Indian Institute of Science, Bangalore - 560012
E-mail: kaushikdas@aero.iisc.ernet.in

Professor, Guidance, Control and Decision System Laboratory Aerospace Engineering Department, Indian Institute of Science, Bangalore - 560012
E-mail: dghose@aero.iisc.ernet.in (corresponding author)

Abstract

This paper addresses the problem of achieving rendezvous in a multi-agent system under various information paradigms. We consider two classes of algorithms (i) Broadcast based algorithms and (ii) Distributed control algorithms. In the first we consider a case where each agent is homogeneous and all agents are controlled by the same broadcast command from a centralized controller. This method has low communication cost. In the second case we consider each agent to implement its own control based on information gathered from its neighbours through a limited sensing capability. In this paper we give a brief overview of the broadcast based methods and some results on the distributed control algorithm where a modification in the decision domain of the agents is shown to yield significant benefits in terms of computational time, when compared with standard algorithms available in the literature. Moreover, we also show its straightforward application to higher dimensional problems which is a considerable improvement over available algorithms in the literature.

1 Introduction

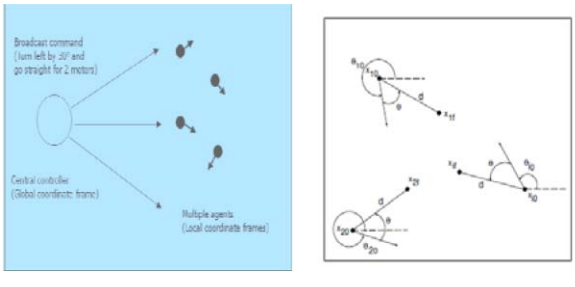
Research on multiple agents is motivated by the fact that instead of using a highly sophisticated and expensive automated agent, it may be advantageous to use a group of small, simple, and relatively cheap autonomous agents. The group of agents can be used to accomplish various tasks in different environments such as tactical operations, exploratory missions, remote monitoring with mobile sensor networks, avoidance of collision and over-crowding in automated air traffic control, cleanups of toxic spills, fire fighting and cooperative search with unmanned air vehicles.

In multi-agent systems, autonomous control of groups of mobile agents which are loosely interconnected through limited range communication links has emerged as a challenging problem. In this paper we discussed two information paradigms; one based on a broadcast mechanism and the other on a distributed control framework. One of the generic tasks that such a system of agent is often called upon to perform is to physically bring all the agents to a common point. This is called a multi-agent rendezvous problem. This problem is important because if rendezvous is feasible, then more general formations are also achievable.

The challenge in these algorithms is to develop control algorithms such that the agents can collectively perform some task. All the agents have limited sensor and communication range and execute some local rule-based strategy depending on the information collected by each agent from the environment, from neighboring agents, and/or from a central controller (in the broadcast case).

2 Broadcast Based Mechanisms

In this framework, each agent executes a command received from a central controller which computes a *common* command for all the agents after observing their positions and orientations [1]. The main advantage of this framework is the saving in communication cost due to the fact that the same command is being transmitted in broadcast mode, as against individual commands to each agent. A possible schematic representation is given in Figure 1.



(a) (b)

Figure 1: (a) Broadcast Mechanism (b) Implementation of the broadcast mechanisms

Note that each agent has a local reference frame which is used by the agent to implement the broadcast command. For instance, if the command is “turn anti-clockwise by 30 degrees and proceed for 3 meters”, then the agent carries out this command in its own local reference frame and not with respect to the global reference frame. Thus, with the

same command issued to all agents, an agent that is oriented at say 10 deg from the global reference frame will attain an orientation angle of 40 deg, while an agent with initial orientation at 110 deg from the global reference frame will attain an orientation of 140 deg (Figure 1:). It was shown in

ISBN 978-9

that this problem can be formulated as a linear programming problem, but the best result of bringing the agents close can be achieved by only one step. In that paper a random perturbation was introduced in each agent's orientation and it was shown that several steps can then be executed with each step leading to a smaller proximity between agents till they converge. Several modifications (based on flag setting to classify behavior of agents) were proposed in [3] and shown to improve convergence.

2.1 Linear Programming Formulation

Let the initial position and initial orientation of the n agents be x (pp) $\square^2$ and $($ $]$, $i0\ i1\ i2$ respectively, for all i i $\{1,...,n\}$. As before, we define the control command to be broadcast as (,. We define our d) performance measure as the half length, denoted by r 0, of the side of a square oriented along the global coordinate frame, and containing all the final positions of the agents. 2

Let this square be centered at z $(1z2$ $\square$.

Assuming that all the agents execute the command (,), d their final positions, given by x $[qq\ i12i]^2\square$ will be,

$$x_{if\ i0\ i0} \quad R \qquad x \quad \begin{matrix} d \\ 0 \end{matrix} R(\ )() \qquad (1)$$

That is

$$\begin{matrix} q_{1\ i1\ i0} & u_{i01} \\ qp_{i2} \end{matrix} \quad \begin{matrix} p \\ \sin \end{matrix} \quad \begin{matrix} \cos \\ i2 \end{matrix} \quad \begin{matrix} \sin \\ \cos \end{matrix} \quad \begin{matrix} \\ i0 \end{matrix} \begin{matrix} \\ u_{i0} \end{matrix} \quad \begin{matrix} \\ 2 \end{matrix} \qquad (2)$$

where, u_1 $d \cos$ and u_2 $d \sin$ are the control variables that replace (,). d Note that the above equations are linear . Now, we formulate the linear programming problem as,

Minimize r

Subject to rp $u \cos$ $u \sin$ z

$$i11\ i02\ i01 \qquad r$$

$$rp \quad u \sin \quad {}_{i21\ i02\ i02} u \cos \quad z$$

i 1, , n.

r 0 The decision vector is (,rz_1, z_2,u_1,u_2) , which the same irrespective of the number of agents remains. Only the number of inequality constraint increases with the number of agents. Also, note that z_1, z_2, u_1 and u_2 are free variables and can take both positive or negative values. An illustration of this process is shown in Figure 2.

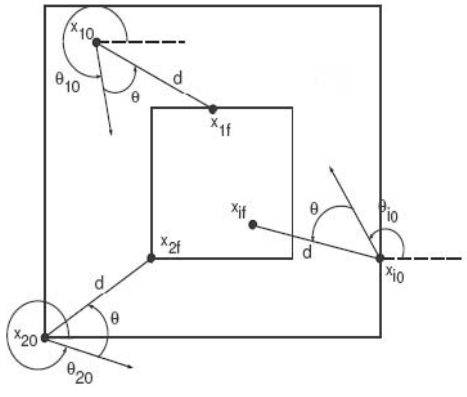


Figure 2: An illustration of how the square reduces in size using the LP solution

2.2 Randomization Mechanisms

2.2.1 Perturbation in the orientation

The solution of the linear programming (LP) problem will yield control instructions which can be broadcast to all the agents using which the agents will move to a new position or within a new square region of smaller area. It can be shown that no further improvement of the performance (that is, reduction in r) can be achieved by repeated use of the algorithm.

We can generalize the process by assuming that each step in the iteration is denoted by the index k , with the first step in the iteration as k 1.

Now, consider a case where, the agents receive a broadcast command containing the LP solution and a command to randomly perturb the final orientation angle after the LP solution has been implemented. In this, the orientation angle, after implementing the LP solution is perturbed by each agent as,

$$\hat{} \qquad {}_{ik,11ik} \quad {}_{ik} \quad {}_{,1}$$

where, the perturbation angle ${}_{ik,1}$ is given by

$$ik,1 \quad i$$

where, ${}_i$ is a random number generated by each agent independently and is a scaling angle which is common to all the agents.

2.2.2 Flag Setting Approach

The main idea behind the flag setting approach is that each agent a_i randomly generates a flag f_i that can take one of

two values. One is high , denoting an active agent and

ISBN 978-9

fh

other is low f_l, denoting a passive agent. The set of agents that have high flags (active) is denoted by R and the set of agents that have low flags (passive) is R_p.

Let the set of active agents at time t be $Rt() \quad \{i \mid f_i \quad f_h\}$ and the set of passive agents at time t be $Rt \{\{1,\} \; f_i \quad f_l\}$ where $i \quad , n$. An active agent is one that will implement the broadcast command and change its state whereas a passive agent will ignore the command and remain stationary. The flag value for each agent may or may not be observed by the central controller (CC). The central controller has information of position and orientation of all the agents (active and passive) at each step. The control command $(\ ,\ )$ computed by the central controller will be broadcast to all the agents, but only the active agents will move.

Let the position and orientation of active and passive agents be $(x_{ak,,pk,pk})$ and $(x\ ,\ )$, respectively. After receiving the broadcast command the new position of the active and passive agents are

$$x_{ak,1} \quad {}^{x}R_{ak,} \quad {}^{R(\ )()} \quad d \quad 0 \quad \text{x} \quad \text{x} \quad {}_{pk,1\ pk}$$

where, $(,)$ is the broadcast control command.

Now, we will propose five different algorithms obtained as combination of the two randomization mechanisms described in the previous section. They are denoted as A ,$(,\ _o)$,$(,\ _N)$, F_o, F_N and are shown in Figure 3. The algorithm A that introduced perturbation only in the angle has been studied in [2].

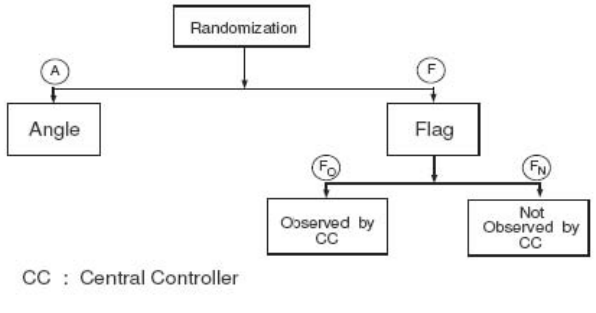


Figure 3: Taxonomy of the randomization mechanism based algorithm

Table 1 : The different performance for the algorithm (Y=Yes; N=No)

Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents
A	Y	N	All	All
$(,\ o)AF$	Y	Y	Active	Active
$(,\ N)AF$	Y	Y	All	Active
oF	N	Y	Active	Active
NF	N	Y	all	Active

2.3 Achieving Positional Consensus at Desired Point

either enters in the sensing range of f

In the previous section, we consider the problem of positional consensus, but did not have control over the point at which the agents can meet.

Suppose we have a pre-specified meeting point then we can achieve this by slightly modifying the previous formulation. In this modified form we define the meeting point as the center of the agent formation and are denoted by $(\ ,\ zz\ )$ Now the inputs to the LP are the initial positions, initial orientation and the meeting point. We can formulate the modified linear programming problem as,

Minimize r

Subject to $\text{r} \quad p \quad u\cos \quad u\sin \quad z \quad r$

$$i11\ i02\ i01$$

$$\text{r} \quad p_i \quad u\sin \quad u\cos \quad z \quad r$$

$$21\ i02\ i02$$

$$i \quad 1, \quad , n.$$

$$r \quad 0$$

The decision vector is $(,ru\ ,u\ )$. Note that the number of decision variables has reduced over the previous formula.

Distributed Control Algorithms

3 Distributed Control Algorithm

The present work proposes a new modified algorithm for distributed implementation of a rendezvous algorithm based on a modified decision domain. Previous work in this area are done by Ando et al. [4] and Lin et al. [5] in two dimensions, where each agent moves toward the rendezvous point by performing a sequence of "stop-and-go" moves. The *stop* mode is basically the *sensing period* and is an interval of fixed length. In the *go* mode the agents will maneuver in an interval of variable length and will move from its current position to a new position. The algorithm proposed by Ando et al. [4] has been extended to arbitrary dimensions by Cortes et al. [6].

ISBN 978-9

In our paper, as in Ando's work, the sensing period is assumed to be zero. Ando's works use algorithms that require determination of the smallest circle that contains a given set of agent positions. These algorithms are called "circumcenter algorithms". Basically it is a special case with dimension 2 of the *miniball* problem [7] in higher dimensions The complexity of this problem is proved to be sub-exponential of order ne [4] in$^{1)}$ $\sqrt{d \ln n}$ $dn \ln$ d dimension of the space with n as the number of agents. So, the number of actual computations is fairly high. In our work we generalize the notion of sensing domain and decision domain and show that by using a rectilinear decision domain the computations can be simplified considerably, thus bringing down the convergence time. We show that our algorithm is far superior in terms of computational time than Ando's algorithm which is the standard algorithm in the literature. The comparison results between RDD and Ando's algorithm in 2D was given in [4]. In this work we extended it to three dimension and the comparative results are also given.

3.1 Preliminaries

Let R_{12} $_2\{,aa$, , $a_n\}$ be the set of robots or agents. The positions of agent a_i is given by p_i $(,x_iy_i)$. The sensor domain of an agent a_i is denoted as S_i and its decision domain is denoted by D_i, where D_i S_i. Information sensed from the decision domain is used to implement the algorithm. Essentially, we introduce the concept that information from the whole of the sensor domain need not be used for decision-making. In Figure 7 we give a schematic of these concepts. Note that, in general, p_i need not be inside S_i.

An agent determines its set of neighbouring agents based on D_i. In this work we assume that the sensing domain (S_i) of all the agents is circular with radius r . The decision domain D_i is a square of side $2d$, with dr /2 $\sqrt{2}$, aligned with the global $(,XY)$ reference frame. This is shown in Figure 7. The set of neighbors of agent a_i is defined as pre-specified

$$N_i \quad \{a_j | (| x_i \quad x_j |) \quad d \text{ and } (| y_i \quad y_j |) \quad d\}.$$

Note that an agent is also its own neighbor, so a_i N_i. Also, if a_j N_i then a_i N_j. In Ando et al. [5] S_i and D_i are the same and are circles.

Let at t_k instant of time the position of agent a_i be $pt_{ik}() \{1(), (), ()\}$. The relative position of agent a_i with respect to its neighbor a_j (say) at time instant t_k is, where $j \{,i, \}$. m

$$zt() \quad p() \quad t_{ijk} \; pt_{ik}()$$

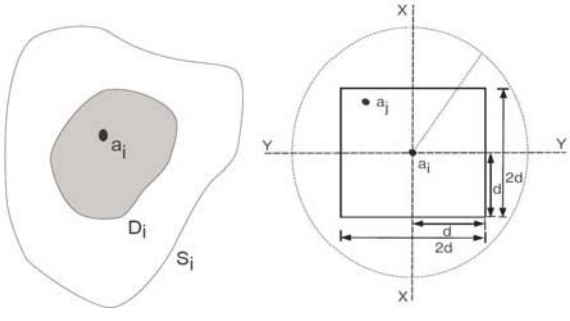
The outcome of this algorithm depends upon the set of neighbors and the relative distance between the neighbor agents. Hence, the position of agent a_i at instant t_k depends upon the set of neighbors, $Nt(\ )$ and relative distance $ik1$ between the agents $zt(\ )$ at time instant $t_{k\ 1}$. So we can write $p(t) = p(t) + U(t)$

ij 1

$ik\ ik \quad 1\ ik \quad 1$

where, $Ut(\ )$ is the control command for agent at $ik1$ instant $t_{k\ 1}$. It is a function of the relative distance and the set of neighbors. Hence we can write

$$Ut() \quad fN(t), z_1(\ ik \quad 1\ ij \quad t_k \quad 1))$$



(a) (b)

Figure 4: (a) General sensor domain and decision domain (b) The sensor domain is circular and the decision domain is a square aligned with the global X Y coordinates.

3.2 Rectilinear Decision Domain (RDD) Algorithm

The RDD (rectilinear decision domain) algorithm retains two basic ideas for a general rendezvous problem (1) Agents who are neighbours remain as neighbours and (2) Agents come closer with each other in some sense till they meet at a point. The RDD algorithm will execute these following steps

1 Each agent *determines* its neighbour set using its decision domain.
2 Each agent constructs the smallest rectangle in two dimensions (or cuboids in three dimensions) aligned with the global coordinate axes that contain all the members of its neighbour set.
3 Each agent computes the centroid of the rectangle (or the cuboids) and moves to it.

To construct a rectangle or cuboids along the global reference frame, an agent requires the information of the maximum and minimum coordinates along any axes, which

is an one dimensional optimization problem over n points. So the computational complexity of this algorithm is n.

Figure 8 illustrates these steps where $ct_{ik}()$ is the centroid of the rectangle at the time instant t_k. These steps are similar to Ando's algorithm [5], but for a few significant differences. In Step 1, Ando's algorithm determines neighbours using the sensor domain S_i. In Steps 2 and 3, Ando's algorithm computes the circumcenter of the neighbours and moves *toward* it subject to a constraint. Unlike RDD which allows the agents to move directly to the centroid, Ando's algorithm may not allow the agents to reach the centroid. These two important differences lead to high computational complexity, and thus higher convergence time, in Ando's algorithm. In RDD, an agent a_i uses the information $P \quad \{(x_j, y_j) | a_{ii} \quad N\}$ where $a_j \quad N_i$, and computes $\max\{x_j\}$, $\min\{x_j\}$, $\max\{y_j\}$ and $\min\{y_j\}$ to obtain the rectangle. The computational complexity of this operation is $() \quad On$

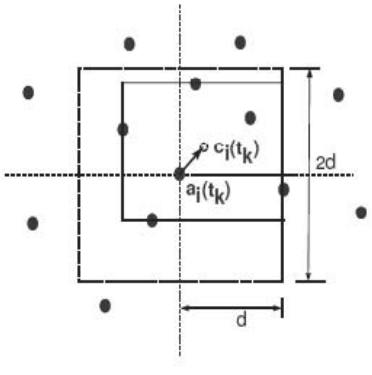


Figure 5: Agents will move to the centroid of the rectangle. The square with broken lines is the decision domain of agent a_i and agent inside it belong to N_i. The solid line rectangle is the smallest rectangle that contains the agent in N_i.

Agent a_i will move to the centroid of the rectangle. The square with broken lines is the decision domain of agent a_i and agents inside it belong to N_i. The solid rectangle is the smallest rectangle that contains the agents in N_i.

It has been proved in [4] that any agent motion will be confined within a square of side d centered at the current position and aligned with the global reference frame. It is also shown in that paper [4] that if any two agents are neighbors at any time instant t_k then they will be neighbor for all time instant tt_k.

3.3 Analysis of Rectilinear Decision Domain (RDD) Algorithm

Let the global convex hull made by the positions of the agents at the time instant t be denoted by $Co\ t$ (). We can define the diameter of the convex hull at the time instant t_k as (()) $max \{ p() \quad p(t) \}$ $dia\ Co\ t_k \{ t_{ik} \}$ where, $ij, \quad \{1,2, \quad, n\}$. When rendezvous is achieved the diameter of the global convex hull is zero. We will first show that the diameter of the global convex hull will reduce at each step.

Let us consider $()$ as the convex hull made by the neighbor set of the agent a_i at the time instant t_k. Let the smallest rectangle containing the neighbor set of agent a_i, and aligned along the global coordinate axes, at time instant t_k be $()$. It is obvious that $Co\ () \quad Rt\ t\ R\ (t)$.

Let $y_{min} \quad \min\{y_j\}$, where y_j is the y coordinate of $a_j \quad N_i$. Other variables are similarly defined. Then we have the following result.

$$mid\ R \qquad ((t)) \quad (1/2)((x_{max} \quad x_{min}), (y_{max} \quad y_{min}))$$

It has been proved in [4] that mid $((Rt))$ and $Co\ (t)$ $((Rt))$ is not a corner point of t ().

Now consider any agent a_i. Let the maximum distance along the X - axis on the right side between agent a_i and its neighbors be d_{lx} and on the left side is d_{ly}. Similarly, along the Y -axis the maximum distance above a_i is d_{ay} and below of a_i is d_{by}. The position of the agent a_i at time instant $(t_{k \quad 1})$ will be

$$xt() \quad x(t) \quad (d \quad d)/2$$
$$yt() \quad yt() \quad (d \quad d)/2$$

The movement of the agent a_i will depend upon $\{d, d, d, d\}$. The agent a_i will be stationary if $d \quad d$ and $d \quad d$. Next we will use a result that $Co\ t_k$ agents at the corner points of global convex hull () cannot remain stationary.

It can be proved that for any agent a_i which is at the corner of $Co\ t$ () and has at least one non-located neighbor, both $d \quad d$ and $d \quad d$ cannot be satisfied.

Now, it is clear that agents at the corner of the convex will move either inside of the convex hull or move along the edges of the convex hull. It has been proved in [4] that $Co\ t\ (\ _{k\ 1}) \quad Co\ t\ (\ _k)$. It is shown in [4] that the diameter

of the global convex hull will reduce at each step, so $((_{k \ 1})$ *dia Co t* $(())$ The above result tells us that the sequence of convex hulls, generated by the positions of the agents will make a descending chain of convex sets.

According to the above results, those agents are at the corner of the global convex hull, RDD algorithm give nonzero control value. The convex hull made by neighbors of agent at time instant is $Co\ ()t \quad \{z_{i}^{a}{}_{kiki}^{t}\}$, where $a_j \quad N_i$. It has been proved that $U(.)$ is not a corner point of Co unless $z_{i1} \quad z_{i2} \quad \cdots \square z_{im} \quad 0$.

Theorem: All the agents will converge to a point as $t_k \quad .$

Proof: $()$ be the convex hull generated by the points $()$ of the $Co\ t_k$. Now any corner $Vt_{k}()$ coincides with the position $pt()$ of, at least, one agent a_i at time instant t_k. Since $Co\ t\ (_{k\ 1}) \quad Co\ t\ (_{k})$, then it is possible to define the convex set Co as the closure of the intersection of all the $()$ for $t_k Co\ t_k 0$.

Let, V be a corner of Co. There exists a sequence of corners $Vt()_{kk}$ of $Co\ t\ ()$, which converges to V as $t_k \quad$. Again, there exists a value $j \quad \{1, \ldots, N\}$, such that $p\ ()_{jk}\ t\ ()Vt$ for infinite number of values of t_k.

Let us assume, without loss of generality, that $j \quad 1$. Now let us select only those values t_l of t_k with $l \quad 0,1, \ldots$ for which $pt_1()_{ll} \quad Vt() \quad$ for all $l \quad 0$.

Since the number of neighbors of agent a_i is non- decreasing as $t_k \quad$. The maximum number of neighbors should be N. As the number of neighbors is finite then we can discard a finite number of values of l and by remembering the index variable we can say that such number is constant (number of neighbors will not change). As the set $\{(), pt\ l \quad 0\}$, where $a_j \quad N$ is infinite set of $_{jl1}$ points and is a compact set then it has at least an accumulation point P_j. Then it is possible [9] to extract a sequence of points converging to P_j.

Let us select only those values of t for which $pt()_{il}$ (where $a_j \quad N_1$) belongs to that sequence and let us rename the variable l by r. This implies that $_{j}()$ converges to P_i as $r \quad$.

Now, $()\ z_{ip}()\ t\ p()$ converges to $P_j \quad P$ $i \quad 1,\ldots, m \quad 1$. Let us assume by slight abuse of notation that $Vt()$ is $_r$ in the sensing region of X_1. We know

$$U(t_{k\ 1}) \quad p(t_{k\ 1}) - p(t_k)$$

From the above equation we can say that $U(t_k)$ goes asymptotically to VX_1, which is a corner of the convex hull because V is a corner of the convex hull. According to that condition, this can not happen unless all the points are at same positions (0) according to those conditions. So $_j$ all the neighbors of agents a_i converge to a point.

Now for any other agents two conditions can arise

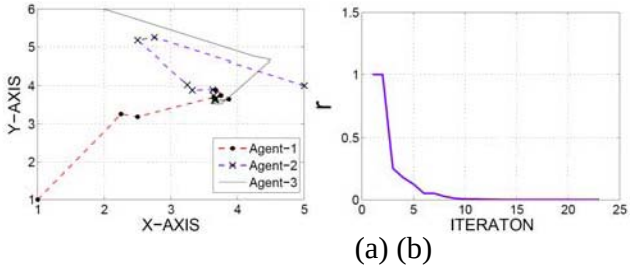
Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents
A	Y	N	All	All
$(\ )AF$	Y	Y	Active	Active

The first case is similar to that given in [9]. So from it we can say that all the agents will converge to a point. The second case is not possible as the initial graph is connected.

4 Simulation Results and Discussions

4.1 Simulations Results

In the first set of simulations we have three agents. Initially we consider $((1,1),45^{o}{}_{o})$, $((5,4),135\ )$, and $((2,6), \quad 45)$as the initial position and orientation angle of the three agents. Using the perturbation technique, with normal distribution for and a scaling angle of 120^{o}, the agents converge to a point after a few iterations (see Figure 9). The convergence criterion for terminating the simulation was when the value of r became less that 210^{4}.



(a) (b)

Figure 6: (a) Trajectory of the agents (b) Reduction in r with iterations

In Figure 10, four snapshots for 10 agents have been shown in two dimensions. Connectedness of the agent placement was defined as follows; two agents are said to be connected if they are within 2 units of each other. Although the initial

placement of the agents was random, we ensured that the agents were connected, which implies that there exists at least one path between any two agents. One can see that all the agents eventually converge to a point. The system converges when the maximum distance between the agents along the global reference frame is less than the decision domain distance , since the agents would converge to a

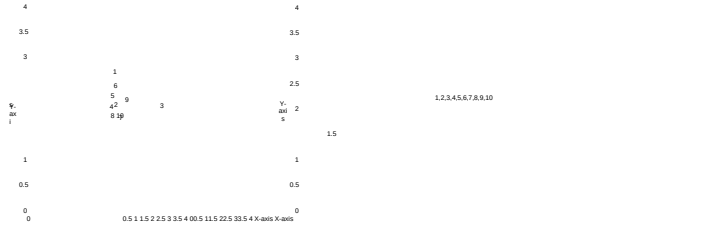
ISBN 978-9

d

single point in the very next step. In Figure 11 we placed 10 agents randomly in three dimensional space. One can see that all the agents converge to a point.

Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents

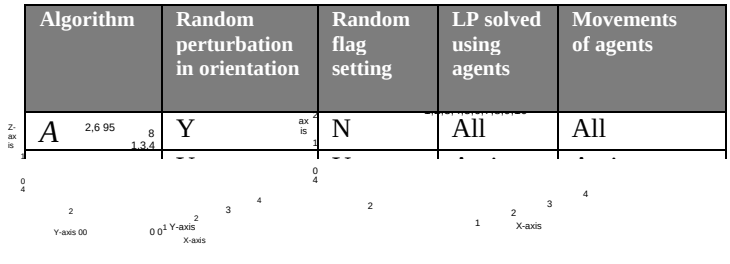
(a) (b)



(b) (d)

Figure 7: 10 agents placed randomly converge to a point in 2D between Ando and algorithm in two dimensions

Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents
A	Y	N	All	All



(c) (d)

Figure 8: 10 agents placed randomly converge to a point

4.2 Comparison Results

The comparative study between Ando's algorithm and the RDD algorithm was carried out in this work. In Table 2, the comparison of computational time to convergence is given. The results show that RDD algorithm is superior in terms of computational time. The average was taken over 25 runs. In Table 3comparison of the number of iterations to converge is given. The iteration number in case of RDD algorithm is a little higher than Ando's algorithm. The reason behind this is the decision domain in case of RDD algorithm is smaller than Ando's algorithm. However, this is easily offset by the fact that RDD takes much less computational time per iteration. In Table 4,5 we made a comparison

between Ando and RDD algorithm in 3D space.

Table 2: Comparison of convergence time between Ando and RDD algorithm in two dimension

Time to Converge

Number of Agents	Ando (secs)	RDD (secs)	Ando/RDD Ratio
20	0.5856	0.0329	17.8065
50	3.0667	0.1758	17.1544
100	12.1486	0.5439	22.3360
150	40.7101	1.7395	23.4031

Table 3: Comparison of number of iterations

Number of Iterations to Converge

Number	Ando	RDD

Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents
A	Y	N	All	All

Table 4 : Comparison of convergence time between Ando and RDD algorithm in three dimensions

Time to Converge

Number of Agents	Ando (secs)	RDD (secs)	Ando/RDD Ratio
20	36.70	0.03	1079
50	543.72	0.12	4531
100	2319.75	0.45	5074
150	4655.62	1.63	2854

Table 5: Comparison of number of iterations between Ando and algorithm in three dimensions

Algorithm	Random perturbation in orientation	Random flag setting	LP solved using agents	Movements of agents
A	Y	N	All	All
$(, o)AF$	Y	Y	Active	Active
$(, N)AF$	Y	Y	All	Active
oF	N	Y	Active	Active
NF	N	Y	all	Active

either enters in the sensing range of f

5 Conclusions

We presented and analyzed a rendezvous algorithm considering a rectilinear decision domain. The computational complexity of RDD algorithm is low compared to the well established Ando's algorithm in the literature. The RDD algorithm is simpler in terms of few computations needed and in relaxing the restriction on the movement of the agents to the centroid of the rectangle.

Acknowledgement

The authors gratefully acknowledge the AOARD/AFOSR for their grant to this project.

References

[1] T. Bretl, Control of Many Agents Using Few Instructions, *Third Robotics Science and System Conference*, Atlanta, 2007.

[2] K. Das and D. Ghose, Positional Consensus in Multi-Agent Systems using a Broadcast Control Mechanism, *American Control Conference*, St. Louis, Missouri, USA, 2009, pp. 5731 - 5736.

[3] K. Das and D. Ghose, Randomization Mechanisms Based Positional Consensus in Homogeneous Multi-Agent Systems, *IISc Centenary International Conference on Aerospace Engineering*, Bangalore, India, 2009, pp 1327 - 1335.

[4] K. Das and D. Ghose, Multi-agent Rendezvous Algorithm with Rectilinear Decision Domain, FIRA, 2010, Bangalore.

[5] H. Ando, Y. Oasa, I. Suzuki and M. Yamashita, Distributed Memoryless Point Convergence for Mobile Robots with Limited Visibility, *IEEE Transactions on Robotics and Automation*, vol. 15, pp 818 - 828, 1999.

[6] Z. Lin, M. Broucke and B. Francis, Local Control Strategies for Groups of Mobile Autonomous Agents, *IEEE Transactions on Automatic Control*, vo. 49, pp 622-628, 2004.

[7] J. Cortes, S. Marti'nez and F. Bullo, Robust Rendezvous for Mobile Autonomous Agents via Proximity Graphs in Arbitrary Dimensions, *IEEE Transactions on Automatic Control*, vol. 51, pp 1289 - 1298, 2006.

[8] B. Gartner: A subexponential algorithm for abstract optimization problems, *SIAM Journal Computers*, 24(5):1018-1035, 1995.

[9] G. Conte and P. Pennesi, The Rendezvous Problem With Discontinuous Control Policies, IEEE Transaction of automatic Control, vol. 55, pp – 279283, 2010.

[10] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. *In Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 1508-1613, Maui, USA, 2003